A Quorum Sensing Inspired Algorithm for Dynamic Clustering

Feng Tan[1] and Jean-Jacques Slotine[2‡]

March 19, 2013

Abstract

Quorum sensing is a decentralized biological process, by which a community of cells with no global awareness can coordinate their functional behaviors based on cell-medium interaction and local decision making. This paper draws inspirations from quorum sensing and colony competition to study the clustering problem.

We propose an algorithm treating each data as a single cell, utilizing the knowledge of local connectivity to cluster cells into multiple colonies simultaneously. The algorithm consists of two stages: first, it spots sparsely distributed "core cells" and determines for each cell its influence radius; second, core cells secrete "auto-inducers" that diffuse into the environment to form colonies. Interactions between colonies eventually determine each cell's identity. We combine the two steps into a dynamic process, which gives the algorithm flexibility to analyze both static and time-varying data.

Finally, we test our algorithm on several applications, including synthetic and real benchmarks datasets, alleles clustering, and dynamic systems grouping and identification. The results suggest that our algorithm performs as well as other cutting-edge methods on static data, while applications on time-varying data like locations of swarms of robots are also promising.

1 Introduction

Quorum Sensing [1] [2] [3] [4] is a decentralized biological process, by which a community of bacteria cells interact with environment locally with no awareness of global information. Each cell secretes auto-inducers that diffuse into local environment and build up concentration. These auto-inducers may be captured by

*[1]Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA fengtan at mit.edu

†[2]Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA jjs at mit.edu

‡*This work was sponsored in part by a grant from the Boeing Corporation

receptors, which can activate transcription of certain genes equipped in the cell. In V. fisheri cells, the receptor is LuxR and the mechanism is shown in Fig.1. When few cells exist in the neighborhood, diffusion kejpg the density of the inducers low, so that no functional behavior will be initiated. However, when the concentration reaches a certain threshold, a positive feedback loop is triggered to secrete more and more auto-inducers and fully activate the receptors. Specific genes are transcribed, and relevant function or behavior expressed by the genes will be performed collectively. We find that cluster analysis in computer science closely resembles this quorum-searching phenomenon. So in this paper we associate these two parts with each other to develop a novel algorithm.

Cluster analysis is to separate a set of unlabeled objects into clusters, so that objects in the same cluster are more similar. Currently, many clustering algorithms have been studied, such as hierarchical clustering (CURE [5], BIRCH [6]), centroid-based clustering (K-means [7]), distribution based clustering (expectation-maximization algorithm), density based clustering (DBSCAN [8]) and spectral clustering (Normalized Cuts [9], Power Iteration Clustering [10]). However, current techniques suffer from several limitations: many algorithms require input of cluster number; sensitivity to outliers and noise often influences clustering results; some algorithms fail to adapt to clusters with different density or arbitrary shape. However, these problems seem to be easily solved in nature by herds of animals, flocks of birds, schools of fish and colonies of cells, whose robustness and flexibility far exceed artificial algorithms. Consequently, it is plausible to learn from nature for new clustering algorithms.

Biological insights can inspire new algorithms [11], like work of Yehuda et al. on borrowing ideas from biology to solve the maximal independent set problem [12]. We think they can also inspire connections between dynamic system control and machine learning algorithms. First, a biologically inspired learning algorithm can be converted into a dynamical process, which suits for real-time control requirements. Second, biological processes handle well with failures and dangers which is similar with the robustness and stability requirements for learning algorithms and dynamic control. Third, biological processes are mostly distributed systems, which may inspire local decision making strategies to control swarms of dynamic systems. In this case bridging dynamic system control with machine learning through biology inspired algorithms is promising.

In this paper, we develop a clustering algorithm inspired by quorum sensing and colony competition. It is not only able to perform clustering on benchmark datasets, but also easy to be integrated with dynamic systems and control strategies. With further extensions made possible through this integration, control theory would be more intelligent and flexible. For the following parts of the paper, we describe details of our algorithm in the second section. Then, we test our clustering algorithm with several experiments. Lastly, we discuss about the results shown in the previous sections, and discuss about future works and extensions.

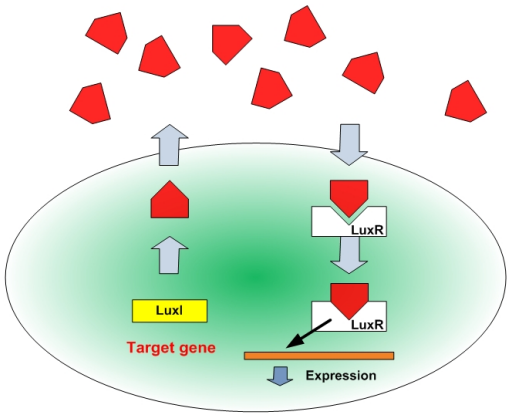

Figure 1: Quorum sensing model

2 Algorithm Model

To design the algorithm, we model the biological process of quorum sensing and colony competitions, including the auto-inducer density model, the local tuning policy, the colony establishments, interactions between and inside colonies and colony splitting and merging processes.

2.1 Gaussian Distributed Density Diffusion

We treat every data as a single cell and use the Gaussian kernel function to describe secretion of auto-inducers as:

$$f(\vec{x}, \vec{x}_i) = e^{-\frac{\|\vec{x}-\vec{x}_i\|^2}{{\sigma_i}^2}} \tag{1}$$

We use Gaussian distribution because it constrains the influence of any single cell in a local region. And the σ_i can be considered as "influence radius" representing the secretion ability of each cell. In such way, we can map all the data into a kernel matrix $M_{n\times n}$

$$m_{ij} = f(\vec{x}_i, \vec{x}_j) = e^{-\frac{\|\vec{x}_i-\vec{x}_j\|^2}{{\sigma_j}^2}} \tag{2}$$

As we know that, $f(x, \vec{x}_i)$ is cell i's influence over environment. So m_{ij} is the influence from cell j to cell i. Moreover, $\vec{d} = M \times \vec{1}_{n\times 1}$ is the vector representing local density of all the cells, where

$$d_i = \sum_{j\neq i}^{n} m_{ij} = \sum_{j\neq i}^{n} e^{-\frac{\|\vec{x}_i-\vec{x}_j\|^2}{{\sigma_j}^2}} \tag{3}$$

If the density of a certain cell is high, we say this cell is "recognized" and located in a well-established colony. Also it will be sensible to set a threshold on m_{ij} to make M sparse, since we only want the cells connected with their local neighbors.

2.2 Local Decision for Influence Radius Tuning

Cells tune their influence radius to connect with neighbors and maintain local density. We design the process to minimize a "hunger factor", which is the error between the density vector and a goal vector as $\vec{a} = a \cdot \vec{1}_{n\times 1}$, then $\vec{a} - \vec{d}$ is the error or the "hunger factor" vector. The "hunger factor" information can be carried with the auto-inducers whose needed input is the local density that can be sensed locally. We use the quadratic error as the cost function:

$$V_{density} = \|a \cdot \vec{1}_{n\times 1} - M \cdot \vec{1}_{n\times 1}\|^2$$

To minimize it, we take the time derivative of $V_{density}$

$$\begin{aligned} \frac{d}{dt}\|\vec{a} - \vec{d}\|^2 &= \frac{d}{dt}(\vec{a} - \vec{d})^T(\vec{a} - \vec{d}) \\ &= -2(\vec{a} - \vec{d})^T \frac{d}{dt}(\vec{d}) \\ &= -2(\vec{a} - \vec{d})^T(\frac{\partial}{\partial \vec{\sigma}}\vec{d})\dot{\vec{\sigma}} \end{aligned} \tag{4}$$

We name the Jacobian matrix as $J = (\frac{\partial}{\partial \vec{\sigma}}\vec{d})$

Then $J_{ij} = \frac{2\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^3} e^{-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^2}}$

Proposition I.

$$\dot{\vec{\sigma}} = J^T(\vec{a} - \vec{d})(*)$$

With this tuning policy for $\vec{\sigma}_i$'s, we can have

$$\frac{d}{dt}V_{density} = -2(\vec{a} - \vec{d})^T JJ^T(\vec{a} - \vec{d}) \leq 0$$

In more details,

$$\begin{aligned} \dot{\sigma_i} &= \sum_{j\neq i} J_{ji}(a - d_j) \\ &= 2\sum_{j\neq i} \frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_i^3} e^{-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_i^2}} (a - d_j) \end{aligned} \tag{5}$$

Here, each term is composed of two parts: the $(a - d_j)$ term represents the "hunger factor" of surrounding cells, and the J_{ji} part represents cell i's potential to satisfy their needs.

However, this proposition is easy to trigger "over-fitting" or get trapped in local minimums. Ill-posed results, such as "super cells" with infinite influence radius, may emerge. On the other hand, it is not feasible for distributed computation like swarms of robots since for any robot to make decisions, it needs complete information of all other agents.

Proposition II.

$$\dot{\vec{\sigma}} = J^T(\vec{a} - \vec{d}) + \beta(M - D)\vec{\sigma} - \alpha\vec{\sigma} + \vec{f}_{init}$$

We regularize the equation in Proposition I. with terms concerning $\|\sigma\|$ and $\sum_{j\neq i}^{n} m_{ij}(\sigma_j - \sigma_i)$. For $\beta(M-D)\vec{\sigma}$, we add the term based on the assumption that cells near each other have similar secreting ability. The D matrix is a diagonal matrix, with the entries $D_{ii} = \sum_{j\neq i}^{n} m_{ij}$. So the ith term in the vector $\beta(M-D)\vec{\sigma}$ equals to $\sum_{j\neq i}^{n} m_{ij}(\sigma_j - \sigma_i)$, which provides diffusive bonding between cells' influence radius in a neighborhood. Also, we add inhibition term $-\alpha\vec{\sigma}$, so that no "super cell" will emerge despite of existing needs, due to dissipation and its own capability, as it is in nature. The $\vec{f_{init}}$ term provides initial actuation to expand the influence radius, which disappears after most of the cells have been connected with their neighbors. It can be regarded as an exploration stage searching local communities. Yet still, this proposition requires global information.

Proposition III.

$$\dot{\vec{\sigma}} = M(\vec{a} - \vec{d}) + \beta(M-D)\vec{\sigma} - \alpha\vec{\sigma} + \vec{f_{init}} \tag{6}$$

We propose the local tuning policy, replacing J^T with matrix M. Here $M(\vec{a}-\vec{d})$ is the local hunger factor vector accumulated at the location of each cell, which is the driving force tuning the $\vec{\sigma}$. When the local environment appears to be "hungry" (local hunger factor is positive), the cell tends to increase its influence radius to satisfy the demand, and vice versa. Our algorithm is more biologically reasonable in this case: the auto-inducers secreted by cells carry the demand information, and spread as the density distribution.

Assume that influence radius in the same colony are mostly similar, then $M \approx M^T$, thus all entries in J^T only adds a proportional term $\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^3}$ before m_{ij}. So intuitively, M should be a good approximation.

Moreover, we use contraction analysis [13] to prove the convergence of both Proposition II and III. Contraction analysis proposes that for the system $\dot{\vec{x}} = f(\vec{x}, t)$, if there exists a constant $\beta > 0$, such that for $\forall\vec{x}, \forall t \geq 0$

$$\frac{1}{2}\left(\frac{\partial f}{\partial \vec{x}} + \frac{\partial f}{\partial \vec{x}}^T\right) \leq -\beta I < 0$$

then all solutions converge exponentially to a single trajectory, independent of the initial conditions. More details with other complex metrics can be referred to [13]. For both propositions, we treat them as $\dot{\vec{\sigma}} = f(\vec{\sigma}) - \alpha\vec{\sigma}$, with the Jacobian matrix $F = \frac{\partial f}{\partial \vec{\sigma}}$. For Proposition II, after rescaling the data such that $\forall i, j, \|\vec{x}_i - \vec{x}_j\|^2 > a^2$

$$\begin{aligned} |F_{i,j}| &= 2\left|\frac{2\|\vec{x}_i - \vec{x}_j\|^4}{\sigma_j^6} - \frac{3\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^4}\right| e^{\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^2}} \\ &\leq \frac{3a}{\|\vec{x}_i - \vec{x}_j\|^2} \leq \frac{3}{a} \end{aligned}$$

Assume after setting a threshold for m_{ij}'s each cell has less than $5a$ neighbors, then $|\sum_j F_{ij}| \leq 15$, so make $\alpha = 15$, we can have $\frac{\partial \dot{\vec{\sigma}}}{\partial \vec{\sigma}} = F - \alpha I$ as a negative diagonally dominant matrix, so that the system is contracting, and converging to a single equilibrium.

For Proposition III, after rescaling the data such that $\forall i, j, \|\vec{x}_i - \vec{x}_j\| > a^2$

$$\begin{aligned} |F_{i,j}| &= |\sum_j m_{ij} \frac{2\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^3} e^{\frac{\|\vec{x}_i - \vec{x}_j\|^2}{\sigma_j^2}}| \\ &\leq \frac{a}{\|\vec{x}_i - \vec{x}_j\|} \leq \frac{1}{a} \end{aligned}$$

Similarly with the less than $5a$ neighbors assumption, $|\sum_j F_{ij}| \leq 5$, make $\alpha = 5$, we have the system for Proposition III contracting, and converge to a single equilibrium.

The convergence proof by contraction analysis is relatively conservative. In our simulations, we can get the system converging to a stable equilibrium with much smaller α choice.

2.3 Colony Establishments and Interactions

In quorum sensing, when the concentration surpasses a predefined threshold, cells begin to produce specific functional genes to perform group behavior. In our algorithm, we use this trait as the criterion for establishing a colony. When the density of a cell belonging to no colony, d_i surpasses a predefined threshold, we establish a new jth colony originating from it and add a $n \times 1$ colony vector $\vec{c}_j$ into the colony matrix C, where $C = [\vec{c_1}, \vec{c_2}, ..., \vec{c_{j-1}}]$, with the only non-zero entry as 1 in the ith term, which is also C_{ij}.
In the Normalized Cuts algorithm [9], which is a cutting edge spectral clustering algorithm widely used for image segmentation tasks, it is designed to minimize the cost function:

$$Ncuts(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(B, A)}{assoc(B, V)}$$

where

$$cut(A, B) = \sum_{i \in A, j \in B} m_{ij}$$

and

$$assoc(A, V) = \sum_{i \in A} m_{ij}$$

We design the colony interactions to minimize a cost function similar to the Normalized Cuts:

$$\begin{aligned} V_{colony} &= \sum_{i \neq j} \vec{c}_i^T (M + M^T) \vec{c}_j - \frac{\gamma}{2} \sum_i \vec{c}_i^T (M + M^T) \vec{c}_i \\ \dot{V}_{colony} &= \sum_{i \neq j} \dot{\vec{c}}_i^{\,T} (M + M^T) \vec{c}_j - \gamma \sum_i \dot{\vec{c}}_i^{\,T} (M + M^T) \vec{c}_i \\ &= \sum_i \dot{\vec{c}}_i^{\,T} (M + M^T)(\vec{c}_e - (\gamma + 1)\vec{c}_i) \end{aligned}$$

Here, $\vec{c}_e = \sum_i \vec{c}_i$ Consequently, by making

$$\dot{\vec{c}}_i = -(M + M^T)\vec{c}_e + (\gamma + 1)(M + M^T)\vec{c}_i$$

$$\begin{aligned} \dot{V}_{colony} &= -\sum_i (\vec{c}_e - (\gamma + 1)\vec{c}_i)^T (M + M^T)^2 (\vec{c}_e - (\gamma + 1)\vec{c}_i) \\ &\leq 0 \end{aligned}$$

We can also transform the interaction equations into a matrix form, where C_e is a matrix with every column same as $\vec{c}_e$:

$$\dot{C} = -(M + M^T)C_e + (\gamma + 1)(M + M^T)C$$

Summing all colony vectors to achieve the environmental colony vector $\vec{c}_e$ also follows the idea of quorum sensing to simplify the calculation through using global variable updates instead of calculating every component. Here all entries in C are saturated in the range of $[0, 1]$. The interactions between colonies are composed of two parts, one part is the mutual inhibitions between colonies, and the other is self-promotion of colonies into the environment. When initially the colonies have not yet been well developed, there is only colony self-expanding, like a neighbor-to-neighbor infection. After the colonies have been expanding for a while, some colonies become neighboring to each other, and the mutual inhibition comes into effect. So finally, it will be a balance between self-expanding and inhibitions from others.

Furthermore, we can explain the interaction in a micro view at the boundary of two competing colonies as shown in Fig.2: suppose that we have two colonies A and B neighboring with each other, with colony vector as $\vec{c}_A$ and $\vec{c}_B$ respectively. For a single cell i in the boundary area between the two colonies, following the interaction rules,

$$\dot{c}_{Ai} = -\sum_j (m_{ij} + m_{ji}) c_{Bj} + \gamma \sum_j (m_{ij} + m_{ji}) c_{Aj}$$

$$\dot{c}_{Bi} = -\sum_j (m_{ij} + m_{ji}) c_{Aj} + \gamma \sum_j (m_{ij} + m_{ji}) c_{Bj}$$

When $\gamma = 1$, it is obvious that $\dot{c}_{Ai} = -\dot{c}_{Bi}$, so if accumulated influence from colony A is larger than from colony B, as $\sum_j (m_{ij} + m_{ji}) c_{Aj} > \sum_j (m_{ij} + m_{ji}) c_{Bj}$, then finally $c_{Ai} = 1, c_{Bi} = 0$, vice versa. Eventually each row in C will have at most one non-zero entry as 1, on the column, whose colony has most accumulated influence towards the cell.
γ is the parameter measuring the relative strength of inhibition and expanding forces: when $\gamma < 1$, the inhibition force is enhanced, there might exist some blank boundaries between colonies. While when $\gamma > 1$, it is easier for the colony factors to spread into neighboring colonies. So at the beginning, it would be wise to tune up γ, to speed up newborn colonies growing, and enhance distributed small colonies merging. Later when colonies have become stable, we can tune γ back to 1 to achieve a distinct clustering result.

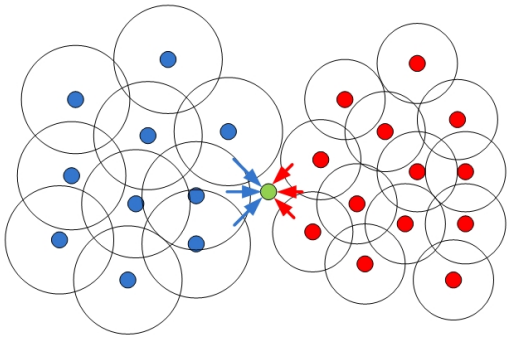

Figure 2: The interactions between two colonies

2.4 Colony Merging and Splitting

Among the established colonies, some may be well connected to each other, while also there may be new colonies. Such scenarios require rules for merging and splitting colony parts. We calculate the ratio between inter colony connections and intra colony connections as a criterion measuring the merging possibility for one colony into another:

$$r_{ij} = \frac{\vec{c}_i^T (M + M^T) \vec{c}_j}{\vec{c}_i^T (M + M^T) \vec{c}_i} \tag{7}$$

We set a threshold, such that if r_{ij} is large enough, colony i is merged into colony j. On the other hand, there may be occasions that new clusters are split from previous colonies. So we set a continuity detecting vector $\vec{s_i}$ for each colony: the evolution of $\vec{s_i}$ follows the same rules of colony interactions. When the detecting process reaches a stable equilibrium, we restart it all over again. Cells identified as outliers in each iteration are marked as "no longer recognized" and become available for forming new colonies upon again.

2.5 Clustering Result

Finally, we get the result by choosing the maximal entry of each row in matrix C, that determines the colony identity of each cell. And cells with null rows are regarded as outliers.
Pseudo Code of the proposed algorithm is presented below:

---

1. Initialize $\vec{\sigma}$ as $\vec{0}$, form the M matrix, set the parameters a, b, β, γ
2. Begin the process:
$\dot{\vec{\sigma}} = M(\vec{a} - \vec{d}) + \beta(M - D)\vec{\sigma} - \alpha\vec{\sigma} + \vec{f}_{init}$
Detect new cluster:
    if $\exists d_i > b (b \leq a)$ and cell i not recognized by any colony
    create a new colony using cell i as core cell
    end
$\dot{C} = -(M + M^T)(C_e - C) + \gamma(M + M^T)C$
$\dot{S} = -(M + M^T)(S_e - S) + \gamma(M + M^T)S$
Cluster segmented detection:
    if in the stable state, $S \neq C$
    update C = S and accept new born clusters
    end
$r_{ij} = \frac{\vec{c}_i^T(M+M^T)\vec{c}_j}{\vec{c}_i^T(M+M^T)\vec{c}_i}$
Cluster merging:
    if $\exists r_{ij} > 0.2, i \neq j$
    then we can merge the colony i into colony j
    end
3. Achieve the clustering results by counting the C matrix

For the parameters choices: γ defines the ability of mutual penetration and crossing density gaps. So with larger γ, colonies are more inclined to merge. β measures similarities of σ_i's in local neighborhood. Larger β would result in smoother σ_i's distribution, yet potentially rugged density distribution. a measures sparsity of the connection graph. With a more connected graph, we tend to have fewer clusters, and vice versa. Hence, we have some basic rules to tune the parameters: if the result suggests fewer clusters than we expect, we can tune down a and γ, and if the influence radius of some cells become too large, we can tune up β and α. In the future, we can design rules tuning parameters dynamically according to our expectations.

3 Experiments

We test our algorithm on several applications, including synthetic and real benchmarks datasets, alleles classifications, and dynamic systems grouping and identification.

3.1 Synthetic Benchmarks Experiments

We provide the clustering results on four difficult synthetic occasions that are nonlinearly separable and follow no specific distribution: the two-chains model, the double-spirals model, the two moons model and the island model. The results are shown in Fig. 3 proving that influence radius tuning ensures that the density distribution closely fits the topology of data, which provides distinct separation boundaries.

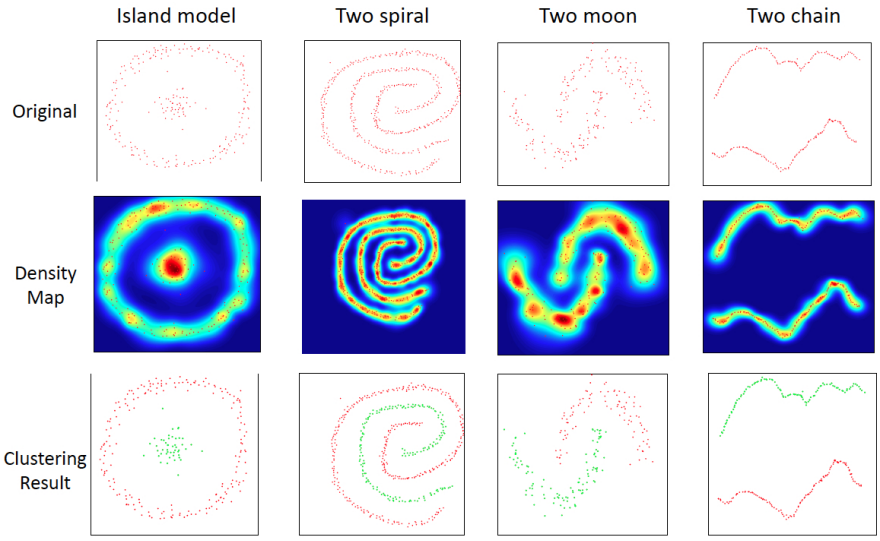

Figure 3: Clustering results on synthetic benchmarks

3.2 Real Benchmarks Experiments

Iris flower dataset
The Iris flower dataset [14], introduced by Sir Ronald Fisher, consists of 150 instances forming 3 clusters, of which two are only nonlinearly separable.
Pendigits dataset
The Pendigits datasets [15] is a multivariate dataset of 10992 instances, each with 16 attributes. We randomly choose 1000 instances for clustering. Also, we build two subsets of the dataset to test: PenDigits01(easier) and PenDigits17(harder), with digits "0", "1" and "1", "7", respectively.
Polbooks dataset
PolBooks [16] is a co-purchase network of 105 political books. Each book is labeled "liberal", "conservative", or "neutral", mostly in the first two category. Here we compare our results with cutting-edge algorithms including Normalized Cuts [9], Ng-Jordan-Weiss algorithm [17] and Power Iteration Clustering [10], shown in Table 1. For the Iris dataset, our performance is comparable to the

cutting edge methods. For the Pen-digits data, we can cluster 10 classes simultaneously with overall correctness rate 86.6% while other methods don't have such ability. Moreover, for the two subcases, we outperform the comparisons. For the network segmentation task of Polbooks dataset, although our algorithm is not designed specifically to solve such tasks, the segmentation result is still very satisfying.

Table 1: Clustering result comparison with NCut, NJW and PIC

Dataset	Instances	Clusters	NCut (%)	NJW (%)	PIC (%)	Ours (%)
Iris	150	3	67.3	80.7	98.0	97.3
Pendigits	1000	10				86.6
PenDigits01	200	2	100.0	100.0	100.0	100.0
PenDigits17	200	2	75.5	75.5	75.5	81.5
Polbooks	105	3	84.8	82.3	86.7	83.8

3.3 Novel Experiment on Application for Alleles Clustering

As introduced in [18], it is very important to understand the similarities of DRB (HLA-DR chain) alleles for the designation of high population coverage vaccines through classifying the supertypes of them. In the work of [18], Wen-Jun et al. analyzed on 559 DRB alleles, and proposed a kernel matrix based on BLOSUM62 matrix measuring the distance between the alleles as

$$D_{L^2}(a,b) = (\frac{1}{\|N\|}\sum_{c\in N}(\hat{K}^3_N(a,c) - \hat{K}^3_N(b,c))^2)^{\frac{1}{2}}$$

So we utilize proposed distance in our Gaussian kernel, with the same tuning policy. The clustering result compared to [18] is shown in Table 2. Our result has no misclassification errors. The only difference is that the 25 alleles classified as outliers have been classified into some clusters in [18] using hierarchical clustering methods. Outliers such as DRB5*0112, DRB1*1525, DRB1*1425, DRB1*1442, DRB1*1469 and DRB1*0832 are discussed as exceptions in their result, which makes their classification more doubtful. Also we share same conclusions on exceptions like DRB1*0338, DRB3*0115; and likewise classify DRB1*1440 , DRB1*1469, DRB1*1477, DRB1*1484, DRB1*14116 and DRB1*14102 into the ST8 supertype.

This experiment proves that our algorithm is effective on clustering multiple clusters simultaneously for alleles clustering data, and also our results support the conclusions of Wen-jun et al.'s work on the mathematical foundation analysis of amino acid chains. The detected outliers may lead further analysis and provide potential directions to biological researchers.

Table 2: Clustering result comparison of alleles clustering

Supertype	Number of Alleles	Misclassified	Outliers
ST52	43	0	0
ST3	63	0	6
ST6	100	0	2
ST8	52	0	2
ST4	93	0	6
ST2	68	0	1
ST5	34	0	1
ST53	6	0	0
ST9	16	0	1
ST7	18	0	1
ST51	15	0	0
ST1	34	0	2
ST10	3	0	3
Overall	25	0	25

3.4 Experiments on Dynamic System Grouping

In this section, we introduce applications on clustering dynamic systems, which show our capability to deal with time varying data in a continuous way. The clustering results that we achieve are flexible and changing according to the distribution of data, which can be regarded as an integral over time.

3.4.1 Application I. Real-time clustering of mobile robots

In natural colonies like fish schools or hordes of buffalos, groups merging and splitting are so smooth and elegant, which gives the colonies flexibility to deal with obstacles and dangers. Our first application is to cluster time-varying data such as locations of working mobile robots imitating the phenomenon. Also as explained in [19], synchronization will enhance the ability to resist noise and improve robustness of the whole network. So potentially, we can couple agents in the same group, to achieve synchronization by forming contracting Lagrangian systems as introduced in [20] and [21]. Such dynamic grouping and coupling would help enhance control performance, which can be analyzed in the future.

In our simulation, 200 robots are located as the previously introduced two-moon dataset, moving around their center locations with radius of 0.5 and random speed. During that time, 30 new robots join the group, and also some robots migrate to form new clusters. We update the distance matrix D and influence matrix M continually. For potential real world application, we can use electromagnetic field emitters, and electromagnetic field intensity sensors to actualize the designed mechanism, replacing the agent-to-agent communications with agent-environment-agent interactions, which is the core idea of quorum sensing.

From video[1] and Fig.5, we can see that the cluster number is first merged down to 3 clusters, and then varies with the merging and splitting events, exactly describing the real-time variation of the robots migration. And shown in Fig.4, the influence radius tuning is capable of handling local density variations: it is tuned down responding to a local high density, and vice versa, to preserve balance. The results prove our capability of dealing with time varying data by using the accumulated information to do dynamic clustering, and handling variations of cluster numbers as well. The results also suggest that using our algorithm to group dynamic systems is feasible and further applications on synchronization and coordination are promising.

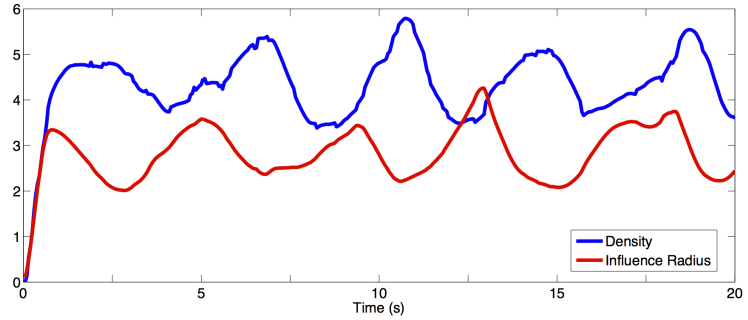

Figure 4: Variation of density and influence radius of a single cell

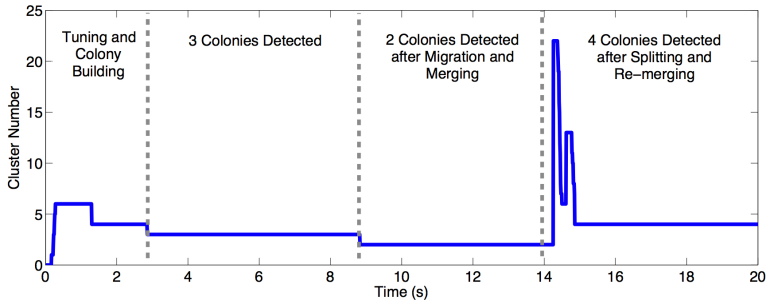

Figure 5: Cluster numbers of over the simulation time

3.4.2 Application II. Multi-model switching

As introduced in [22] [23], multi-model switching control could improve the transient control performance and precision. Here we propose a new method for multi-model switching control:

[1] http://www.youtube.com/watch?v=EshxTGNpQC4

Suppose we have a system with unknown parameters, however we know that there are limited possible choices of parameters configuration.

1. Initially, we use adaptive control to assure acceptable performance. Simultaneously, we have tens of virtual systems simulating with the same control input whose parameters scatter around the pre-known choices.

2. When the density of virtual system is stable after tuning, we calculate local density of the real system:

$$d_r = \sum_i^n e^{-\frac{\|\vec{f}_r - \vec{f}_i\|^2}{\sigma_i^2}} \tag{8}$$

   where $\vec{f_r}$ and $\vec{f_i}$ are Fourier transform vectors of the input of the real system and virtual systems.

3. If d_r exceeds a predefined threshold, we know the real system belongs to a virtual cluster. Hence we can get the real parameters and switch to robust control.

4. Further, if the parameters vary again, by detecting d_r dropping down, we can resume adaptive control and wait for the next time that d_r surpasses the threshold.

For experiment, we use 60 virtual dynamic systems as $m_i\ddot{x}_i + b_i|\dot{x}_i|\dot{x}_i + k_i x_i = u_i, m_i, b_i, k_i$ are unknown constants. The parameters m_i, b_i, k_i are scattered around three known choices: $[4,3,2], [2,4,3]$ and $[3,2,4]$, with 20 systems each. And we have a "real" system, whose parameters m_r, b_r, k_r are set as [4,3,2] initially, and then changed to [2,4,3]. Assume the task is to track the trajectory of $x_d(t)$. According to adaptive control theory, we need control law:

$$u = \hat{m}(\ddot{x}_d - \lambda(\dot{x} - \dot{x}_d)) + \hat{b}|\dot{x}|\dot{x} + \hat{k}x - k_1 s \tag{9}$$

where $s = \dot{\tilde{x}} + \lambda\tilde{x}$, $k_1 > 0$ is a constant, and adaptive law:

$$\dot{\hat{m}} = -s(\ddot{x}_d - \lambda(\dot{x} - \dot{x}_d))$$

$$\dot{\hat{b}} = -s|\dot{x}|\dot{x}$$

$$\dot{\hat{k}} = -sx$$

Since we know $\tilde{m}(\ddot{x}_d - \lambda(\dot{x} - \dot{x}_d)) + \tilde{b}|\dot{x}|\dot{x} + \tilde{k}x$ converges to 0 asymptotically and thus $u \approx m(\ddot{x}_d - \lambda(\dot{x} - \dot{x}_d)) + b|\dot{x}|\dot{x} + kx$, systems with similar parameters require similar inputs. So that we can use the input Fourier transform vector to measure distance between systems in our algorithm.

As we can see from the simulation results Fig.6 and Fig.7, soon after the multi-model switching starts at $t = 10$ seconds, the density of the real system surpasses the threshold 5, and the parameters are estimated correctly. After the parameters change at $t = 20$ seconds, sharply the density drops along with the

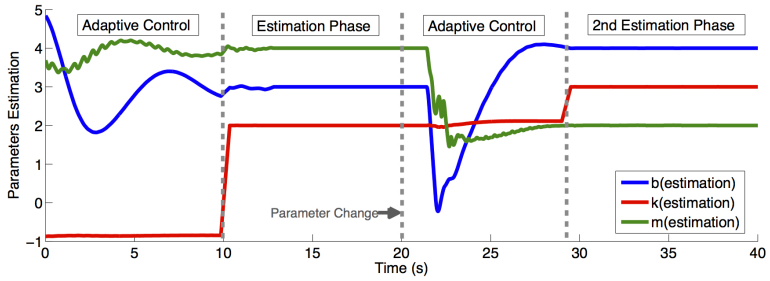

Figure 6: Parameter estimations of the real system

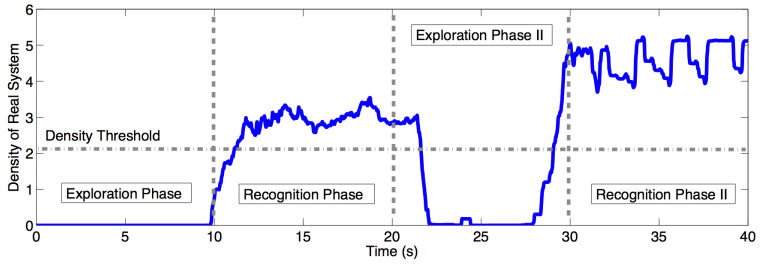

Figure 7: Density of the real system

control mode switched to adaptive control. After about another 10 seconds, the density is high again, and the system is correctly estimated with new parameters.

With the applications above, we show the potential of combining our algorithm with dynamic systems. The algorithm attempts to imitate the smooth grouping and coordination of natural colonies and the results prove the reliability of proposed algorithm.

4 CONCLUSIONS

This paper presents our dynamic clustering algorithm inspired by quorum sensing as a potential bridge between machine learning and control. Experiments on different types of unlabeled datasets show that our algorithm performs as well, if not better, as some cutting edge clustering methods on static datasets and performs well on dynamic clustering tasks. Our algorithm's advantage over existing algorithms can be concluded as:

1. Since the influence radius is tuned to preserve local connectivity, we can adapt to clusters with different sizes and variations. We can cluster nonlinearly separable data that follow no specific distribution, and handles noise and outliers well.

2. Our decentralized algorithm is potentially suitable for parallel and dis-

tributed computation. Potential applications on real mobile robots would replace agent-agent communications with agent-environment interactions.

3. We can cluster multiple colonies simultaneously. And the segments are dynamically adjusted through smooth merging and splitting.

4. The algorithm is dynamic, which is easy to be combined with real mechanical systems. Novel control that is more flexible and robust theory for coordinating complex systems may emerge.

The overall computation complexity would be $O(n)^3$ with single processor. However, if we use the algorithm on real robots clustering, with distributed computation, the computation of single robot would be hugely reduced to linear time.

For future work, first, we will conduct further research developing rules dynamically tuning the parameters. Second, for dynamic systems' metrics, we may need various better methods for extracting feature vectors rather than just analyzing the fast Fourier transform vector of the inputs. Third, we will look into the possibilities of using our clustering algorithm on much more applications involved with dynamic systems. And last but not least, we will develop new control theories that utilize the information gained from our novel algorithm, to further improve control performance and synchronization with more natural self-organizing coordination.

References

[1] M.B. Miller and B.L. Bassler. Quorum sensing in bacteria. *Annual Reviews in Microbiology*, 55(1):165–199, 2001.

[2] C.M. Waters and B.L. Bassler. Quorum sensing: cell-to-cell communication in bacteria. *Annual Review of Cell and Developmental Biology*, 21:319–346, 2005.

[3] T.D. Seeley and P.K. Visscher. Group decision making in nest-site selection by honey bees. *Apidologie*, 35(2):101–116, 2004.

[4] S.C. Pratt. Quorum sensing by encounter rates in the ant temnothorax albipennis. *Behavioral Ecology*, 16(2):488–496, 2005.

[5] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. volume 27, pages 73–84. ACM SIGMOD Record, 1998.

[6] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases, 1996.

[7] J.A. Hartigan and M.A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[8] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. volume 1996, pages 226–231. AAAI Press, 1996.

[9] Shi Jianbo. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[10] F. Lin and W.W. Cohen. Power iteration clustering. *ICML (to appear)*, 2010.

[11] S. Navlakha and Z. Bar-Joseph. Algorithms in nature: the convergence of systems biology and computational thinking. *Molecular Systems Biology*, 7(546), 2011.

[12] Y. Afek, N. Alon, O. Barad, E. Hornstein, N. Barkai, and Z. Bar-Joseph. A biological solution to a fundamental distributed computing problem. *Science's STKE*, 331(6014):183, 2011.

[13] W. Lohmiller and J.J.E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[14] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[15] F. Alimoglu and E. Alpaydin. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 637–640 vol. 2. IEEE, 1997.

[16] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[17] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[18] W.J. Shen, H.S. Wong, Q.W. Xiao, X. Guo, and S. Smale. Towards a mathematical foundation of immunology and amino acid chains. *Arxiv preprint arXiv:1205.6031*, 2012.

[19] N. Tabareau, J.J. Slotine, and Q.C. Pham. How synchronization protects from noise. *PLoS Computational Biology*, 6(1):e1000637, 2010.

[20] Q.C. Pham and J.J.E. Slotine. Stable concurrent synchronization in dynamic system networks. *Neural Networks*, 20(1):62–77, 2007.

[21] S.J. Chung and J.J.E. Slotine. Cooperative robot control and concurrent synchronization of lagrangian systems. *IEEE Transactions on Robotics*, 25(3):686–700, 2009.

[22] K.S. Narendra and J. Balakrishnan. Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, 42(2):171–187, 1997.

[23] K.S. Narendra, J. Balakrishnan, and M.K. Ciliz. Adaptation and learning using multiple models, switching, and tuning. *Control Systems, IEEE*, 15(3):37–51, 1995.