Using Geometric Diffusions for Recognition-Primed Multi-Agent Decision Making

Xiaocong Fan
The Behrend College
The Pennsylvania State University
Erie, PA 16563, USA
xfan@psu.edu

Meng Su
The Behrend College
The Pennsylvania State University
Erie, PA 16563, USA
mengsu@psu.edu

ABSTRACT

Several areas of multi-agent research, such as large-scale agent organization and experience-based decision making, demand novel perspectives and efficient approaches for multiscale information analysis. A recent breakthrough in harmonic analysis is diffusion geometry and diffusion wavelets, which offers a general framework for multiscale analysis of massive data sets. In this paper, we introduce the "diffusion" concept into the MAS field, and investigate the impacts of using diffusion distance on the performance of solution synthesis in experience-based multi-agent decision making. In particular, we take a two-dimensional perspective to explore the use of diffusion distance and Euclidean distance in identifying 'similar' experiences–a key activity in the process of recognition-primed decision making. An experiment has been conducted on a data set including a large collection of battlefield decision experiences. It is shown that the performance of using diffusion distance can be significantly better than using Euclidean distance in the original experience space. This study allows us to generalize an anytime algorithm for multi-agent decision making, and it also opens the door to the application of diffusion geometry to multi-agent research involving massive data analysis.

Categories and Subject Descriptors

I.2.0 [**Artificial Intelligence**]: General—*Cognitive simulation*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Design, Experimentation, Performance

Keywords

Cognitive agent, Recognition, Decision Making, Diffusion distance, Experience

1. INTRODUCTION

Several areas of multi-agent research demand the investigation of novel perspectives and efficient approaches for multiscale information analysis. For instance, in large-scale agent organization [13, 15], it is often desirable for an agent to construct a global picture of the ever-evolving communities at different hierarchical levels in order to optimize its decisions on community formation or task coordination. In social network analysis [11, 5, 6], multiple agents often need to work together to identify similar events or patterns of events buried in a huge information space that may cross policy boundaries (e.g., global terrorist network), where multiscale information analysis is the key to effective reasoning at multiple policy/priority levels.

In particular, cognitive agents empowered with recognition-primed decision making capability [10, 4] also rely on multiscale information analysis. Given a new decision situation, these agents, typically under time-stress, need to recall decision experiences that worked before in a situation 'similar' to the current situation, and synthesize a workable solution out of those similar experiences. Here, due to the nature of time pressure, one challenge is the design of effective anytime algorithms for identifying similar experiences from an experience base where the data are often large-scale, high-dimensional, and distributed. Multiscale information analysis is certainly a viable approach to regulate an agent's inference by levels: refining a solution at a more resource-demanding level only when time permitted.

A recent innovation in multiscale harmonic analysis is diffusion maps and diffusion wavelets [1, 2], which generalizes the techniques of harmonic analysis in Euclidean space to massive data sets on manifold. Diffusion maps provide a natural low-dimensional embedding of high-dimensional data that is suited for subsequent tasks such as visualization, clustering, and regression. It is also shown that diffusion wavelets and multiresolution scaling functions can be constructed efficiently, which can then be exploited for compression and denoising of functions and for clustering and learning [3]. The potentials of this diffusion framework have been demonstrated in several areas including classification [1], multiscale/hierarchical representation of gene array data and document corpora (e.g. [14]).

In the diffusion framework, diffusion distance $d_m(x, y)$ is a key geometric quantity that measures the rate of connectivity of the points x and y by paths of length m in the data. Unlike the geodesic distance, this metric is robust to perturbations on the data. This motivates us to conjecture whether diffusion distance can be effectively exploited by cognitive agents to identify similar experiences and conduct multi-scale experience recognition in decision making.

The objective of this research is three-fold. First, the concept of diffusion distance as an intrinsic geometric measure

Cite as: Using Geometric Diffusions for Recognition-Primed Multi-Agent Decision Making, X. Fan and M. Su, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 275-282

is introduced into the MAS field. Second, by applying this concept to a realistic data set (experiences for battlefield decision making), we would like to understand the effectiveness of using diffusion distance to identify past experiences that are "similar" to a given decision situation. Third, we explore the influence of using Gaussian kernel as the local affinity and the potential impacts of this kernel.

The rest of the paper is organized as follows. We briefly introduce the diffusion geometry framework in Section 2. We frame our problem in Section 3, and describe our two-dimensional approach to investigating the usefulness of the notion of diffusion distance in Section 4. In Section 5, we present the experiment and give result analysis. Some generalization considerations are provided in Section 6, and Section 7 concludes the paper.

2. GEOMETRIC DIFFUSIONS

The diffusion geometry framework [1, 9], a recent innovation in harmonic analysis and spectral graph theory, generalizes some aspects of the Newtonian paradigm, in which local infinitesimal transitions of a system lead to global macroscopic descriptions by integration. This framework introduces the notion of diffusion distance (and diffusion maps, diffusion wavelets) and offers a general foundation for multiscale analysis on massive data sets.

Given a massive data set, diffusion geometry starts with the premise that a similarity measure of nearby data points can be meaningfully defined. For a data set X with n observations, suppose a pairwise similarity matrix $W = \{w_{i,j}\}$ can be built. The $n \times n$ matrix W is called a kernel function, representing some notion of affinity or similarity between pairs of points of X. One can think of the data points as being the nodes of a symmetric graph whose weight function is specified by W. The kernel W defines the local geometry of X, and it is sparse because $w_{i,j}$ is typically reset to 0 if it is below a predefined threshold. Although the choice of W is both data- and goal-dependent, a typical choice is the Gaussian kernel: $w_{i,j} = e^{-(||x_i - x_j||/\varepsilon)^2}$, where ε is a scale (precision) parameter of the Gaussian distribution.

W is then normalized to obtain a matrix $P = D^{-1}W$, where D is the diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} w_{i,j}$. The matrix P is called the diffusion operator on X, where each entry $p(i,j) = w_{i,j}/D_{ii}$ can be viewed as the transition probability of going from node i to node j in one time step of the Markov chain on X. Since P encodes the geometric information about the data set X, the transitions directly reflect the local geometry defined by the immediate neighbors of each node in the graph of the data.

Carrying this "random walk" view further, P^t gives the probability of transitions from one node to another in t steps. As one runs the walk forward, the local geometry information is being propagated and accumulated (integrated), leading to a global characterization of the data set. As t becomes higher, the node-connectivity is diffusing further away to cover more and more neighbor points. From a machine-learning perspective, the powers of P allow the incremental discovery of clusters separated by connection-bottlenecks, and each cluster defines a region from which, for any point in the cluster, the probability of escaping is very low.

A family of diffusion distances D_t at step t is defined as

$$D_t(x,y)^2 \triangleq \sum_{z=1}^{n} (p(z,t|x) - p(z,t|y))^2 / w(z) \tag{1}$$

where $p(z,t|x)$ is the probability of the random walk from node x to node z after t steps, and $w(z) = \sum_x w_{z,x}$.

The diffusion distance $D_t(x,y)$ reflects the connectivity of the data at scale $2t$. It involves summing over all paths of length t connecting x to y (and y to x); two points are closer, if there is a large number of short paths between them. As a consequence, $D_t(x,y)$, unlike the Euclidean or geodesic distance, takes into account all evidences relating x and y, and is very robust to noise perturbation. This makes it especially fit for experience-based agent decision making, where we could employ diffusion distance to design inference algorithms based on the majority of preponderance.

Diffusion distances can be computed using eigenvectors $\{\psi_k\}(0 \le k \le L)$ and eigenvalues $\{\gamma_k\}$ of P [2] where $1 = \gamma_0 > |\gamma_1| \ge |\gamma_2| \ge \cdots |\gamma_L|$:

$$D_t(x,y)^2 = \sum_{k=1}^{L} \gamma_k^{2t} (\psi_k(x) - \psi_k(y))^2 \tag{2}$$

A family of diffusion maps $\{\Psi_t\}_{t \in \mathbb{N}}$ is defined by

$$\Psi_t(x) \triangleq \begin{pmatrix} \gamma_1^t \psi_1(x) \\ \gamma_2^t \psi_2(x) \\ \vdots \\ \gamma_L^t \psi_L(x) \end{pmatrix}. \tag{3}$$

The components of $\Psi_t(x)$ are the diffusion coordinates of x at the scale t. The diffusion map Ψ_t embeds the data X into an Euclidean space, such that in this space, the Euclidean distance is equal to the diffusion distance (relative to certain accuracy) [2], or equivalently,

$$D_t(x,y) = \left\| \Psi_t(x) - \Psi_t(y) \right\|. \tag{4}$$

That is, the ordinary Euclidean distance in the diffusion space measures the intrinsic diffusion distance on the data. While this well establishes the relationship between the Euclidean distance and diffusion distance, they are different metrics in nature. Euclidean distance often fails in capturing the global spatial-relation among points of a massive data set, while the diffusion distance has a global meaning for data sets with a nonlinear geometric structure (manifold), and is very robust to noise data.

Since in many practical applications the spectrum of the matrix P has a spectral gap with only a few eigenvalues close to 1 and all the others much smaller than 1, the diffusion distance at a large enough scale t can be well approximated by only the first few δ eigenvectors, with a negligible error of the order of $O(\gamma_{\delta+1}/\gamma_\delta)^t)$. Such an observation, together with Equation (4), provides a theoretical justification for dimensional reduction. Hence, appropriately selected eigenfunctions of Markov matrix P lead to macroscopic representations at different scales. In particular, the top eigenfunctions permit a low-dimensional geometric embedding of the original data set.

From the diffusion map concept, Coifman and Maggioni [3] further introduced the notion of diffusion wavelets, generalizing the construction of classical wavelets to discrete data clouds. They proposed a multiresolution analysis process for constructing the bases of orthonormal scaling functions and wavelets. This opens the door to the application of the diffusion framework to the 'parameter-free' analysis of hyper-dimensional massive data sets. For instance, the idea of diffusion wavelets has been used to facilitate the clustering of document corpora at different levels [3, 14].

Table 1: Feature-based situation description of example experiences

Target-specific						Situation-specific		Weather		
Insurgent	*IED*	*Crowd*	*Level*	*Speed*	*CloseToRoute*	*UnitReadiness*	...	*PrecipitationType*	*Rate*	*Visibility*
Yes	No	No	XHigh	Slow	Yes	85	...	Rain	High	Haze
No	No	Yes	Low	Fast	Yes	80	...	Hail	Light	Fog
No	Yes	No	High	None	No	60	...	Snow	High	Fog

3. THE PROBLEM DEFINITION

Naturalistic decision making (NDM) focuses on how people actually make decisions in realistic settings that typically involve ill-structured problems, uncertain dynamic environments, shifting/competing goals, and time stress [16]. One particular model is Klein's Recognition-Primed Decision framework (RPD) [7]. The RPD model is based on the supposition that in complex situations human experts usually make decisions based on the recognition of similarities between the current decision situation and previous decision experiences. Cognitive studies have shown that over 95% of human decisions conform to the RPD model in various time-stressed situations [8].

The RPD model [7] has a recognition phase and an evaluation phase. In the recognition phase, a decision maker synthesizes the observed information about the current decision situation into appropriate cues or pattern of cues, then employs a strategy called "feature-matching" to recall experiences worked before in a *similar* situation. The similar experiences are then used to construct candidate solutions, each of which is a course of action that might be applicable to the current situation to achieve the goal under concern. In the evaluation phase, the RPD model stresses on Simon's satisficing criterion [12] rather than objective optimization: a decision maker considers the candidate solutions one by one, terminating the evaluation as soon as a workable solution is obtained. Due to the dynamic and uncertain nature of the environment, a decision maker can misinterpret the situation in the recognition phase, which could lead to unexpected events happening when the committed course of action is implemented. Hence, it is necessary that the decision maker, while in the evaluation phase, keeps monitoring the situation as it evolves in order to further diagnose the recognition and construct new solutions if time permitted.

Several computational RPD models have been proposed and built into cognitive agent architectures (e.g., [10, 4]). Among others, one challenging issue is, given the timing constraints associated with a decision task (e.g., 5 seconds), how to design an effective anytime algorithm such that the situation can be recognized reasonably well and the best possible solution(s) can be constructed in due time. Going deeper, the problem can really be reduced to the investigation of an effective feature-matching approach where experiences with higher and higher quality (similarity) can be identified in successive recognition cycles as far as time allows.

We thus can frame our problem as follows. Suppose an agent has a large set E of experiences (knowledge collected from domain experts) about decision making on a certain task type, and the set $F = \{F_j | 1 \leq j \leq m\}$ of features (type of information) relevant to this task type is fixed. Each experience $e_i = \langle S_i, A_i \rangle \in E$ has two parts: feature-based situation description $S_i = (f_1, f_2, \cdots, f_m)$ where f_j is a value for feature $F_j (1 \leq j \leq m)$, and a course of action $A_i = (a_1, a_2, \cdots, a_\theta)$ (i.e. a solution successfully implemented before in a situation as described by S_i).

Table 2: Course of action and related parameters

Seq	*Action*	*Description*
1	AgentRecall(Squad, C1)	Recall C1 number of squads
2	AgentAssignment(Squad,C2)	Assign C2 number of squads
3	AgentRecall(EOD, C3)	Recall C3 number of EODs
4	AgentAssignment(EOD, C4)	Assign C4 number of EODs
5	ReadinessRecover(C5, C6)	Recover C6 number of agents' readiness to C5 percentage
6	MoveTo/RushTo/CreepOver	C7: get to a target's location
7	CaptureInsurgent/Monitor /DisperseF/DisperseW /DisperseP/RemoveIED	C8: how to handle a threat

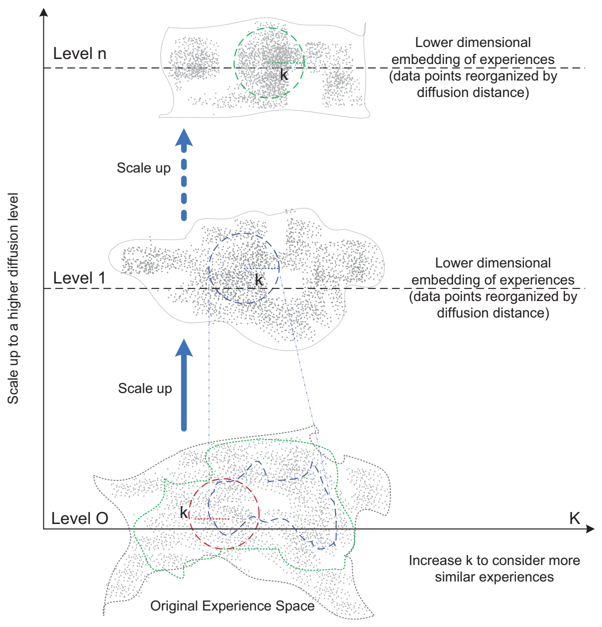
Given a new decision situation $D = (d_1, d_2, \cdots, d_m)$ where d_j is a value for feature $F_j (1 \leq j \leq m)$, the feature-matching problem is to find a set $E^* \subset E$ such that those experiences in E^* are similar to D in terms of all the features being considered. The solution construction problem is to synthesize a new course of action such that the solution part of each experience $e_i \in E^*$ has an appropriate influence on the new synthesized solution. Our objective here is to examine whether the use of diffusion distance, as compared with the traditional Euclidean distance, can help an agent to find the set E^* with a higher quality so that a better solution (course of action) can be synthesized for D afterward.

Experiences in real-world problems are high-dimensional data. For this study, we choose to use a data set E including 16,383 decision making experiences about how C2 teams react to potential threats that emerge unexpectedly in a metropolis. Members of a C2 team, consisting of an S2 agent (intelligence cell) and an S3 agent (operations cell), need to work collaboratively to handle three types of targets: crowds, insurgents, and IEDs (Improvised Explosive Device). Two types of friendly units are under S3's charge: Squad units and EOD (Explosive Ordnance Disposal) teams. The S3 agent, when making decisions on how to handle a specific threat, needs to consider information about 34 features (which are classified as target-specific, situation-specific, or weather-related), and decide resource allocation actions appropriate for that threat.

Table 1 gives a portion of the situation description of three example experiences, and Table 2 gives the fixed set of action types and the corresponding parameters. All the experiences in E are complete (both the situation description part and the course of action part are available). Since the courses of action of all the experiences in E have the same sequence as shown in Table 2, all that matters are the values for parameters $C1$ through $C8$. For each experience $e_i \in E$, the values of $C1$ through $C8$ can be concatenated into one string, which will be referred as the label of e_i below.

4. METHODOLOGY

We take a two-dimensional approach to examine whether



Level n
Lower dimensional embedding of experiences (data points reorganized by diffusion distance)
k
Scale up
Level 1
k
Lower dimensional embedding of experiences (data points reorganized by diffusion distance)
Scale up to a higher diffusion level
Scale up
Level O
k
K
Increase k to consider more similar experiences
Original Experience Space

Figure 1: The Two Dimensional Approach

diffusion distance can work better than Euclidean distance in identifying similar experiences in a hyper-dimensional space.

The idea is illustrated in Figure 1. Along the horizontal dimension, as the value of the parameter k (i.e., the k-nearest-neighbor approach) is increased, more and more 'similar' experiences can be considered in solution construction. We would like to find out whether the quality of constructed solutions is in any way related to the number of similar experiences considered. If the correlation were positive, we could design an anytime algorithm such that an agent can construct a higher quality solution by considering a larger range of similar experiences when time permitted.

Along the vertical dimension, we start with applying Euclidean distance to the original experience space for similar experience identification. We then scale up to a diffusion level i, applying diffusion distance to the "transformed" experience space at level i for similar experience identification. At the diffusion level i, each point is a lower-dimensional embedding of the corresponding point at a lower diffusion level $j (j < i)$ (down to the original space). Moreover, each point is re-located to a different position as the diffusion level changes. In other words, all the data points in the original space have been re-organized as the diffusion process goes. This means that the set of k nearest neighbors (wrt a given point) collected at level i can be very much different from the set of k nearest neighbors collected at level $i-1$. For instance, in Figure 1 if the set of k nearest neighbors at level 1 were mapped back to the original space (level O), the points might scatter beyond the range occupied by the k nearest neighbors in the original space. In the original space, there can be some noise points in the set of k nearest neighbors, say, due to the inappropriate use of distance metrics. It is noted that diffusion maps can filter out high-frequency noises, which suggests that noises can be reduced as diffusion level increases. This raises an interesting question: can we take advantage of this behavior implied by the diffusion process to design an anytime algorithm for solution construction in recognition-primed decision making?

4.1 Forward Embedding

Given a set X of hyper-dimensional data points, diffusion maps (and scaling functions, diffusion wavelets) can be computed to capture and embed the data points in X into lower-dimensional spaces. One challenge is, given a new data point $q \notin X$ that was not considered in computing the diffusion maps, how to take advantage of the already computed diffusion maps to study the property of q at different diffusion levels? We call this the forward embedding issue. Unlike the traditional semi-supervised learning, a central problem in forward embedding is how to efficiently embed (represent) a new data point q into a diffusion space, so that the embedding of X in that space can be exploited to study q.

Wang and Mahadevan [14] faced the same issue in their study of multiscale analysis of document corpora. However, this issue becomes trivial due to their use of term-term matrix instead of the document-document matrix [3] as the diffusion operator. This allows them to compute the multiscale embedding of a new document at any diffusion level j by simply applying the extended basis functions at level j to the new document represented as a term-vector. In general, given a data set $X \in \mathbb{R}^{n\times m}$, where m is the dimensionality of the feature space, and n is the number of data points. The first thing is to decide which dimensionality, n or m, to keep in constructing the diffusion operator P. Keeping n suffers from the issue of scalability: typically m is fixed when a problem is defined, but n scales up as new data are collected. However, keeping m, which is used by Wang et al, is only applicable to limited situations. First, m ought to be reasonably large (e.g. hundreds); if it is small, the benefit of diffusion approach can be compromised. Second, the constructed diffusion operator should be meaningful as far as the problem is concerned. In Wang's case, the term co-occurrence relationship as captured by the term-term matrix is a meaningful kernel for document analysis.

For our experience data set, the dimensionality of the feature space is only 34. In order to position a new decision situation at different diffusion levels (so that we can use the nearest neighbors to label the new situation), we cannot avoid the forward embedding issue. Here we take a simple approach, leaving the issue open for future studies. Given a new point q, we first get its λ number of nearest neighbors in the original space. These λ points, called binder points, will be used as a boundary to confine the location of the new point at a diffusion space. In order to label the new point, first get the k nearest neighbors for each of the λ binder points, then use majority vote to label the new point by considering the labels of all the $\lambda \times k$ points.

5. EXPERIMENT

As explained in Section 3, the data set E used in this experiment includes $16,383$ decision making experiences, which are represented in a 34-dimension feature space. Since the value ranges of the 34 features are different (some are indicator variables, some are percentages, some are integers with fixed ranges), the data set is first standardized such that all the features have the same range $[0,1]$. We denote this standardized set by $X_{n\times m}$, where $n = 16383$, $m = 34$.

As stated in Section 2, from the set X, we built a symmetric matrix W with Gaussian weights $w_{i,j} = e^{-(||x_i - x_j||/\varepsilon)^2}$, where $w_{i,j}$ is the similarity between points x_i and x_j. W

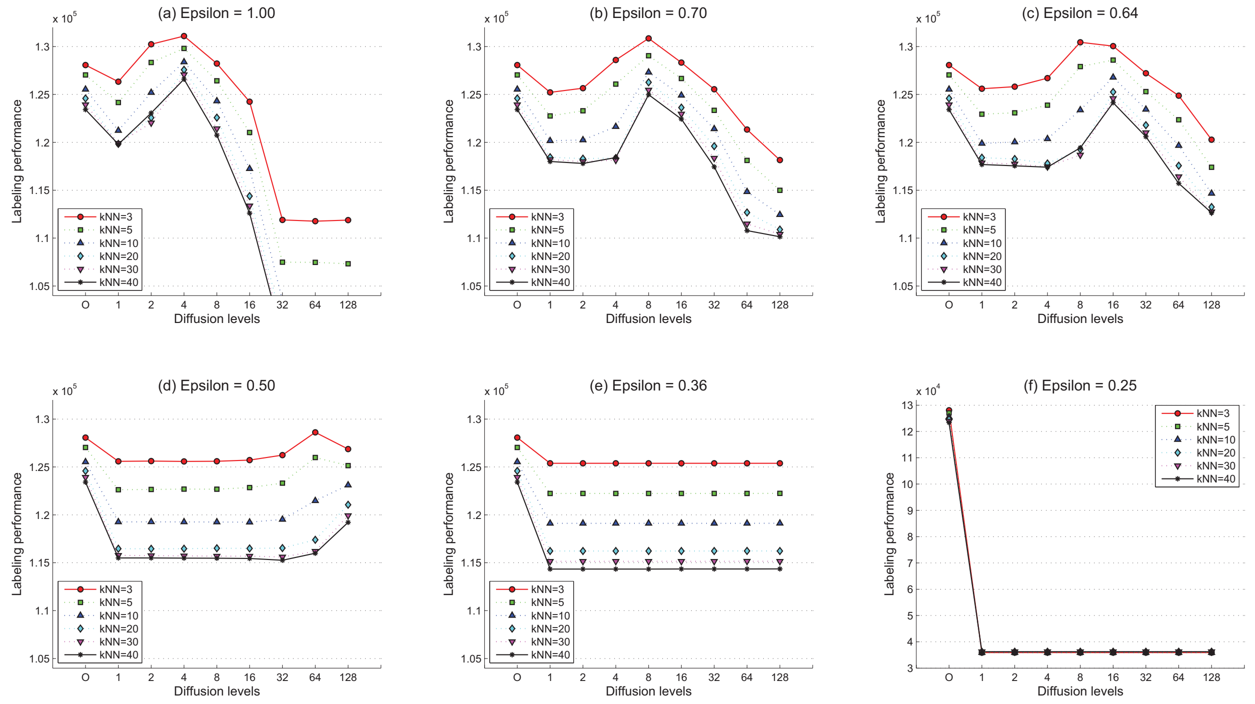


Figure 2: The comparison of labeling performance using diffusion maps

can be taken as a graph, where points $x_i, x_j \in X$ are connected by an edge of weight $w_{i,j}$. Following [3], instead of using $P = D^{-1}W$ as the diffusion operator (see Section 2), we normalized W to build a symmetric diffusion operator $T = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$, which is a similar matrix of P. The powers of T constitute an object of interest for the study of the geometric structures of X at various scales (diffusion levels). For this study, applying Eq. (3), we constructed diffusion maps at 8 levels: 1, 2, 4, 8, 16, 32, 64, and 128.

Also, it is commented [3] that, when the Gaussian kernel is used, the choice of ε, the standard deviation which measures the magnitude of the local similarity, can greatly affect the diffusion effectiveness. Because ε heavily depends on the nature of the data set being studied, we took ε as one control variable to explore how it may affect the performance of solution construction. We varied ε from 1.0, 0.70, 0.64, 0.50, 0.36, to 0.25. Thus, in total, we obtained $6 \times 8 = 48$ diffusion maps for the set of decision experiences X.

The performance was evaluated in terms of the recoverability of labels. In particular, suppose we are given a decision situation D together with its label ζ_D. After a set $E^* = \{e_1, e_2, \cdots, e_k\}$ of k similar experiences are identified for D, we can generate another label ζ_D^* for D by majority vote: each part ($C1$ through $C8$) of ζ_D^* is determined by majority vote of the corresponding part of experiences in E^*. The score of ζ_D^* is the weighted sum of the correctness (0 or 1) of each part as compared with the corresponding part of the known label ζ_D.

5.1 Labeling Performance on Diffusion Maps

We first applied the 48 diffusion maps to each experience in the set X itself, where along the horizontal dimension as illustrated in Fig. 1 we varied the parameter k (kNN from 3, 5, 10, 20, 30, to 40). Figure 2 plots the solution construction (labeling) performance under various configurations, where we use 'O' to refer to the performance in the original Euclidean space.

The first surprise is the impact of k (the number of nearest neighbors being considered). It is not that the more neighbors considered, the better solution can be constructed. Before this experiment, we conjectured that an anytime algorithm could be such designed that an agent would take more 'similar' experiences into consideration, if time permitted, to synthesize a better solution incrementally. The experiment result (Fig. 2 (a-f)) indicates the opposite. Regardless of the changes of diffusion level or ε (except the extreme case when $\varepsilon = 0.25$), the performance dropped significantly as kNN increased from 3 to 40. A reasonable explanation to this surprise is that, when more neighbors are considered, the chance of considering noises becomes higher and higher, which degrades the solution being constructed.

The second finding is quite encouraging: the performance of using diffusion distance (levels 1-128) can be significantly better than using Euclidean distance in the original space (level O). For instance, in Fig. 2(a), the level-1 performance was lower than level-O performance, but the performance increased considerably to its peak as the diffusion level increased from 1 to 4 (regardless of the value of kNN). It is also worth noting that the performance started dropping as the diffusion level increased from 4 on, even lower than level-O performance when the diffusion level was larger than 16. We can find the same pattern from Fig. 2 (b-d), only that the diffusion levels when peak values occur are different.

The key question is, why the use of diffusion distance at

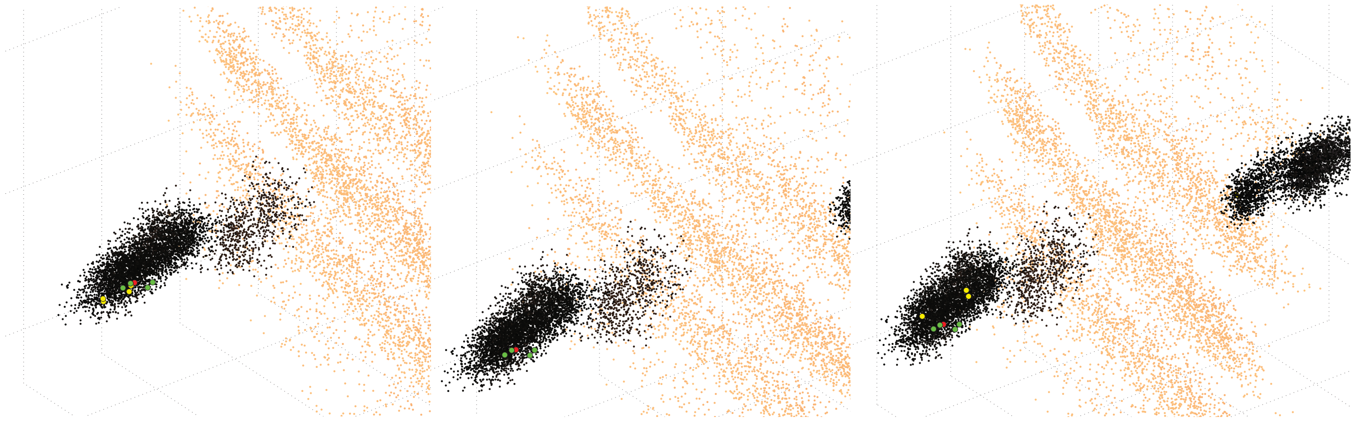


Figure 3: Data points plotted in diffusion spaces according to the Eigenvectors 3, 4, and 5 of the diffusion operator T. The nearest neighbors of a data point p (marked in red) change as the diffusion level changes. The four nearest neighbors of p at diffusion level 4 are marked in green; its four nearest neighbors at diffusion level 1 and 64 are marked in yellow. (a) Data set is deployed at the level-1 diffusion space (magnitude 10^{-2}); (b) Data set is deployed at the level-4 diffusion space (magnitude 10^{-4}); (c) Data set is deployed at the level-64 diffusion space (magnitude 10^{-34}).

different diffusion levels could so greatly affect the quality of identifying 'similar' experiences (consequently, the labeling performance)? The reason lies with the data re-organization power of diffusion distance. As described in Section 2, by definition, the diffusion distance at scale t between points x and y reflects their connectivity in the graph of the data: it is small if there is a large number of short paths of length t connecting x and y (i.e., a large probability of transition from x to y and vice versa). As a consequence, unlike the geodesic distance, diffusion distance is robust to noise perturbation [2], and the data with higher proximity tend to move closer after the re-organization as the scale t increases. The reason why the performance dropped after the peak is also due to the re-organization of data. As the scale t increases, the measurement becomes coarser and coarser, resulting in smaller and smaller difference among nearby points. Consequently, after the peak, it is very likely that more and more noise points cluster around a point q and overwhelm q's real neighbors. This re-organization power of diffusion distance is shown in Fig. 3, where the collection of nearest neighbors of a data point keep changing as the diffusion level changes.

This seems to suggest an anytime algorithm. An agent could use Euclidean distance to produce a base solution. The agent then starts another round of computation, if time permitted, to identify 'similar' experiences by applying diffusion maps until the level with peak performance.

Interestingly, the result also indicates a wave (ripple) effect as ε changes. In particular, the diffusion level bearing the peak performance changed from 4, to 8, 16, and 64 as ε changed from 1.0, to 0.70, 0.64, and 0.50. In other words, the peak performance occurred at a higher diffusion level when ε was smaller. This suggests that in time-stressed decision making, an agent could adjust ε to expedite the process of finding better solutions.

From Fig. 2, it is not only that the parameter ε could affect the occurrence time of the peak performance. Fig. 2 (e-f) also indicate that ε, if chosen inappropriately, can produce a performance significantly worse than the traditional Euclidean approach.

5.2 Applying Diffusion Maps to New Data

In real-time decision making, it is highly likely that an agent has to deal with decision situations that it cannot find any exactly matching experiences. We used another data set Y ($Y \cap E = \emptyset$) containing 5899 experiences to examine whether the diffusion maps learned from E could be applied to novel situations in Y.

The experiment result on Y gave us the same patterns as shown in Fig. 2. In particular, (1) Regardless of the changes of diffusion level or ε (except the extreme case when $\varepsilon = 0.25$), the performance dropped significantly as kNN increased from 3 to 40; (2) The performance of using diffusion distance (levels 1-128) can be significantly better than using Euclidean distance in the original space (level O); and (3) It indicates a wave (ripple) effect as ε changes: the peak performance occurred at a higher diffusion level when ε was smaller. Such a pattern-conformance feature can actually be exploited to design a two-phase algorithm, which is to be discussed in the next section.

As discussed in Section 4.1, for data set Y we are also interested in the impact of λ. In Figure 4, for each ε, we plot the agent's labeling performance at different λ settings (Note that those were the performance when kNN=3, the best possible performance as kNN varied. See Fig. 2 for the pattern as kNN varied). Also note that here only the peak performance matters, because an anytime algorithm could be such designed that an agent won't consider refining its solution beyond the diffusion level that bears the best performance. In Fig. 4 (a), for example, the best performance for all λ settings occurred at diffusion level 4.

Figure 4 tells us two things. First, it is not the case that more binder points resulted in better performance. For instance, the performance when $\lambda = 20$ was not the best at any ε setting, and it actually was the worst when $\varepsilon = 0.50$. The setting of $\lambda = 7$ always produced the best, or close to the best, performance. Second, the best performances produced by different λ settings are not significantly different. For instance, the best performances produced when $\lambda = 3$

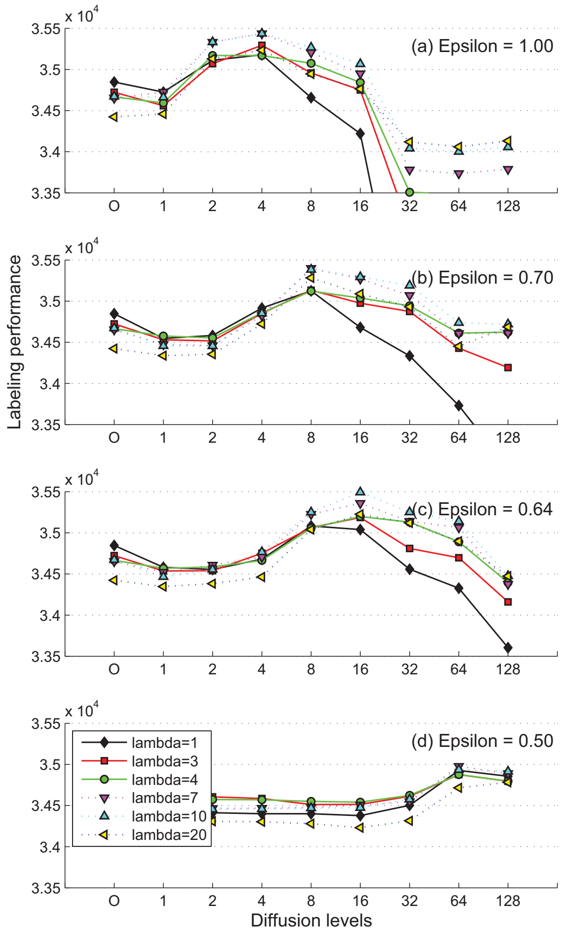


Figure 4: Applying diffusion maps to testing points

and when $\lambda = 4$ were almost the same in each ε setting.

However, the time (on average) it took for an agent to construct a solution increased significantly as λ increased. For this experiment, the computing time changed from 2.1s, to 3.87s, 5.51s, 8.56s, 12.1s, and 23.25s, as λ changed from 1, to 3, 4, 7, 10, and 20. In a time-stress environment (like battlefield), oftentimes an agent needs to help a human operator make a decision in no more than 10 or 15 seconds. Thus, λ should be reasonably small; depending on the time-stress level, it is acceptable to set λ to a number in [3, 7].

6. GENERALIZATION

We now generalize from our experiment to design an anytime algorithm to be used by cognitive agents for experience-based decision making. Since the construction of diffusion maps is time-consuming, we propose a two-phase approach: an offline part (unsupervised learning) where an agent computes diffusion maps (with different settings for ε and diffusion level) and elicits heuristics for appropriate parameter settings, and an online anytime algorithm where the agent applies the diffusion maps to new decision situations and fine-tunes its recognition performance whenever time permitted. This approach is motivated by our finding of pattern-conformance: the performance when a collection of diffusion maps are applied to the data set from which those maps were constructed, has the same pattern (trend) when this set of diffusion maps are applied to a new data set (through the forward embedding strategy given in Section 4.1). The pattern-conformance actually can be explained by the "locality" assumption of the diffusion geometry [1]. According to the continuity of the manifold structure, the diffusion distance should be continuous with respect to the local perturbation in Euclidean distance. Hence, it is reasonable to confine a new data point by its binder points.

```
$[\varepsilon^*, \kappa^*, \mathrm{Tau}, \mathrm{Map}]$ = **GetMapHeuristics**(X)
/*$[M]_n$ means $M$ is an array of size $n$ */
1. kNN = $[\kappa_1, \kappa_2, \cdots, \kappa_k]$;
2. EP = $[\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_m]$;
3. DL = $[\tau_1, \tau_2, \cdots, \tau_n]$;
4. $[PO]_k$ = getEuclideanPerformance(X,kNN);
5. $[\mathrm{Map}]_{m\times n}$=computeDiffusionMaps(X,EP,DL);
6. $[\mathrm{PeakTau}]_{m\times k}$, $[\mathrm{PeakVal}]_{m\times k}$=getPeaks(X,kNN,Map);
7. $\varepsilon^* = \mathrm{argmax}_m[\mathrm{PeakVal}]_{m\times k}$;
8. $\kappa^* = \mathrm{argmax}_k[\mathrm{PeakVal}]_{m\times k}$;
9. Tau = GetArrayOfGoodTaus(Map, $\varepsilon^*, \kappa^*$);
10. MaxDiffPerf = PeakVal($\varepsilon^*, \kappa^*$);
11. If MaxDiffPerf > $PO(\kappa^*)$;
12.     return $[\varepsilon^*, \kappa^*, \mathrm{Tau}, \mathrm{Map}]$;
13. Else
14.     return null
```

```
**AnyTimeRefiner** $(X, D, \mathrm{Map}(\varepsilon^*), \kappa^*, \mathrm{Tau}, RCT, T)$
/*$T$: total time allowed*/
/*$RCT$: an estimation of the computation time per round*/
1. $\Lambda = [1, \lambda_2, \cdots, \lambda_j]$;  $\lambda = getNext(\Lambda)$;
2. EuPerf = getEuclideanPerformance($X, D, \kappa^*$);
3. Perf = EuPerf;
4. While $((T \geq RCT) \wedge \mathrm{Tau}.hasNext())$
5.     $\tau = getNext(\mathrm{Tau})$;
6.     Perf$'$ = getDiffPerformance(Map($\varepsilon^*, \tau$), $D, \kappa^*, \lambda$);
7.     Perf = $max$(Perf, Perf$'$);
8.     $T = T - RCT$;
9. End While
10. Perf = $max$(EuPerf, Perf, Perf$'$);
11. While $((T \geq RCT) \wedge \Lambda.hasNext())$
12.     $\lambda = getNext(\Lambda)$;
13.     Perf$'$ = getDiffPerformance(Map($\varepsilon^*, \tau$), $D, \kappa^*, \lambda$);
14.     Perf = $max$(Perf, Perf$'$);
15.     $T = T - RCT$;
16. End While
```

Figure 5: The algorithms for solution construction.

The two-phase approach is given in Fig. 5. GetMapHeuristics() returns the constructed diffusion maps and three heuristics: the setting of ε and the number of kNN which produced the best peak performance, and a set of diffusion levels, at which the performance were better than the Euclidean performance. For example, from Fig. 2, we can set $\varepsilon = 1.0$, kNN=3, and Tau = $[2, 4, 8]$. Of course, the heuristics derived from other data sets can be very much different.

Note that GetArrayOfGoodTaus (line 9 of GetMapHeuristics) actually computes a set of diffusion levels (e.g. Tau = $[2, 4, 8]$), instead of just the level produced the best performance (e.g. 4), and this set is to be considered in Any-

TimeRefiner(). The reason is that the performance corresponds to each diffusion level is just a statistical measure; for a specific data point, its best solution could be constructed at a level different from the level produced the best "overall" performance.

Every time an agent gets a new decision situation D (together with a deadline T for making the decision), it executes AnyTimeRefiner(), exploiting those heuristics obtained offline. In the first part (lines 4-9), the agent, sticking with just one binder point for D, considers the diffusion levels in Tau one by one, if time permitted, to construct a better solution for D. In the second part (lines 11-16), the agent considers the λ settings in Λ (i.e. number of binder points), if time permitted, to get an improved solution.

This approach could be further extended in several ways. First, as more and more new decision experiences are accumulated, an agent can improve its solution construction performance by periodically update the diffusion maps to reflect the new information. Second, diffusion maps are built on a diffusion operator, which depends on the kernel matrix W being used. Since a given kernel only captures a specific feature of the data set, its choice should be guided by nature of the given problem. Although the Gaussian kernel, as used in this study, are suggested for many problems [3], it might be inappropriate for certain data sets. It is worthwhile to explore the use of other kernels for performance improvement. Third, as explained before, when $\lambda = 4$, getDiffPerformance() took 5.51s to construct a solution for a new decision situation. This time, of course, depends on the size of the diffusion map being used (in this study, it embedded 16,863 points). The time could be greatly reduced when a multi-agent approach can be employed. For instance, the data set under concern can be split into multiple parts, thus multiple smaller diffusion maps, and each agent only works on a fraction of the original map. When a new decision situation comes, the agents first negotiate to decide who will take the responsibility. This enables the agent group to handle more parallel decision tasks and quicker.

In addition, the diffusion wavelets approach [3] is very efficient for automatic multiscale clustering. Our next step is to apply the diffusion wavelets approach to multi-scale decision making where the collection of decision experiences are distributed among multiple cognitive agents.

7. CONCLUSION

People can typically switch among multiple recognition scales to understand a complex situation. Likewise, cognitive agents simulating/supporting human users generally need to construct multiple resolutions (cognition and metacognition) of the overwhelming amount of sensing information before taking an appropriate course of action. Diffusion geometry is a recent breakthrough in multiscale harmonic analysis. This general framework based upon diffusion processes exactly meets the need for efficient approaches to multiscale analysis of massive data sets.

Our contribution is two-fold. First, we took a two dimensional perspective to explore the use of diffusion distance and Euclidean distance in identifying 'similar' experiences– a key activity in the process of recognition-primed decision making. We conducted an experiment on a data set including a large collection of battlefield decision experiences. It is shown that the performance of using diffusion distance can be significantly better than using Euclidean distance in the original experience space. From the study we generalized an anytime algorithm that can be used by cognitive agents for time-stressed decision making. The potential impact of using Gaussian kernel as the local affinity was also discovered.

Second, the concept of diffusion distance as an intrinsic geometric measure is introduced into the MAS field. As shown in this study, diffusion distance and the whole diffusion geometry framework will demonstrate its vigor in many agent research areas that involve massive data analysis.

8. REFERENCES

[1] R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc. of Nat. Acad. Sci.*, 102(21):7426–7431, 2005.

[2] R. R. Coifman and S. Lafon. Diffusion maps. *Appl. Comput. Harmon. Anal.*, 21(1):5–30, 2006.

[3] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Appl. Comput. Harmon. Anal.*, 21(1):53–94, 2006.

[4] X. Fan and J. Yen. R-cast: Integrating team intelligence for human-centered teamwork. In *Proceedings of AAAI'07*, pages 1535–1541, 2007.

[5] M. E. Gaston and M. desJardins. Agent-organized networks for dynamic team formation. In *Proceedings of AAMAS'05*, pages 230–237, 2005.

[6] R. Glinton, K. P. Sycara, and P. Scerri. Agent organized networks redux. In *Proceedings of AAAI'08*, pages 83–88. AAAI Press, 2008.

[7] G. A. Klein. Recognition-primed decisions. In W. B. Rouse, editor, *Advances in man-machine systems research*, volume 5, pages 47–92. Greenwich, CT: JAI Press, 1989.

[8] G. A. Klein. Recognition-primed decision making. In *Sources of power: How people make decisions*, pages 15–30. MIT Press, 1998.

[9] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Advances in Neural Information Processing Systems 18*, pages 955–962. MIT Press, 2005.

[10] E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings of AAMAS'04*, pages 202–209, 2004.

[11] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of AAMAS'02*, pages 475–482, 2002.

[12] H. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[13] J. Vazquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *AAMAS*, 11(3):307–360, 2005.

[14] C. Wang and S. Mahadevan. Multiscale analysis of document corpora based upon diffusion models. In *21st International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[15] Y. Xu, E. Liao, P. Scerri, B. Yu, M. Lewis, and K. Sycara. Towards flexible coordination of large scale multi-agent teams. In *Coordination of Large-Scale Multiagent Systems*, pages 287–309, 2006.

[16] C. E. Zsambok and G. Klein, editors. *Naturalistic Decision Making*. Lawrence Erlbaum Associates, 1997.