Group Formation Among Decentralized Autonomous Agents

Elth Ogston, Maarten van Steen, and Frances Brazier
Department of Computer Science,
Vrije Universiteit Amsterdam

Abstract

This paper examines a method of clustering within a fully decentralized multi-agent system. Our goal is to group agents with similar objectives or data, as is done in traditional clustering. However, we add the additional constraint that agents must remain in place on a network, instead of first being collected into a centralized database. To do this we connect agents in a random overlay network and have them search in a peer-to-peer fashion for other similar agents. We thus aim to tackle the basic clustering problem on an Internet scale, and create a method by which agents themselves can be grouped, forming coalitions. In order to investigate the feasibility of this decentralized approach, this paper presents simulation experiments that look into the quality of the clusters discovered. First the clusters found by the agent method are compared to those created by k-means clustering for two-dimensional spatial data points. Results show that the decentralized agent method produces a better clustering than the centralized k-means algorithm, placing 95% to 99% of points correctly. A further experiment explores how agents can be used to cluster a straightforward text document set, demonstrating that agents can discover clusters and keywords that are reasonable estimates of those identified by the central word vector space approach.

I. Introduction

Agents that wish to cooperate within a multi-agent system must have a means of finding each other. The straightforward solution to this problem is to create a central directory server that is able to match requests. However, this centrally directed solution limits the autonomy of agents with respect to their choice of partners, and it limits the scalability of the multi-agent system as a whole. Ideally agents would, on their own, be able to group together to form cliques of like minded agents. As a result they would know their potential partners (members of their social circle) and could directly negotiate new partnerships based on more information than a directory server would contain. Grouping agents in this way, based on similar objectives, can be viewed as a clustering problem. Clustering has been studied in a variety of fields, notably statistics, pattern recognition and data mining. These fields have a wide range of purposes in mind, for instance discovering trends, segmenting images, or grouping documents by subject. However, in all of these disciplines the underlying problem is the same; given a number of data items, create a grouping such that items in the same group are more similar to each other than they are to items in other groups [4]. Most algorithms for clustering focus on how to form these groups given a file or database containing the items. Yet, for Internet applications like finding similar

web pages or finding agents with similar interests, items can be widely distributed over many machines and the issue of collecting the items in the first place gains importance. Centralized clustering is problematical if data is widely distributed, data sets are volatile, or data items cannot be compactly represented. Decentralization, on the other hand, is a thorny problem. Even in the centralized case where each data item can be compared to every other data item, perfect clusters can be hard to find. Decentralization creates the additional complication that even if a correct classification can be determined with the incomplete information available, the location of items belonging to a class also needs to be discovered.

This paper considers the case where classification is straightforward and focuses on the question of finding potential cluster members in a decentralized fashion. By studying in depth a simplified example of agent grouping we hope to gain insight into dynamics that can be used to create more complex, self-organizing agent communities. With this purpose in mind, we view clustering as a search problem in a multi-agent system in which individual agents have the goal of finding other "similar" agents. Agents aim to form groups among themselves, and these groups constitute a clustering. In large scale Internet systems potentially millions of agents are spread across possibly as many machines. As a result each agent will always have a view of only a very small fraction of the rest of the system. Our research is concerned with the minimal abilities and resources required by such agents. We create an abstract model of simplified agents which have a very small range of actions and act using straightforward decision functions. Furthermore, these agents can generally only communicate in a peer-to-peer manner with a limited amount of additional coordination among small groups.

We study the behavior of this abstract system through simulation experiments. In the first of these experiments agents are each given a two-dimensional spatial point and seek to group themselves based on the Euclidean distance between their points. In this setting the quality of the clusters produced can be measured. A second experiment considers more complex data for which similarity measures are more subjective. Each agent is given a text document and agents try to find groups based on document similarity. In both experiments agents are initially randomly assigned a small number (5) of neighbor agents. These neighbors are an agent's only view of the system as a whole. Based on these local views, agents form clusters with the closest points they come across. Agents within a cluster coordinate, combining their local views to allow each member to search a broader range of neighbors for better matches. Clusters are limited in

size by a user-defined parameter. Once clusters have grown to this size they spilt when better matches are found by their members, allowing stronger new clusters to develop. We show that for the two-dimensional data case agents produce better clusterings then the standard k-means algorithm. We further find that agents can also discover reasonable clusters for high dimensional text data, even when similarity is imprecise.

The remainder of this paper first discusses the application of the clustering problem to multi-agent systems and surveys previous work in Section II. Section III then sketches the model we study, followed by a precise description of the simulated procedure. Section IV presents our methodology and experimental results. A discussion of directions in which these experiments can be expanded in future concludes the paper.

II. Background And Related Work

Middle agents or directory services are commonly used in multi-agent systems to enable the location of agents with particular capabilities [1]. Such services, however, add an essentially centralized component to an ideally decentralized agent world. The formation of groups of agents based on like interests provides a potential alternative for very large decentralized systems where maintaining a directory becomes too costly. Such groups place potential partners for collaboration in an agent's immediate local environment [3]. When agents' interests include jointly working on common tasks this process evolves into coalition formation; the negotiation of agreements between agents with complementary skills for the distribution of work and rewards [7] [11] [12]. Clustering, as studied in this paper, is a more basic problem, yet an essential component of the process of forming coalitions. A multi-agent system made up of heterogenous agents that cannot somehow identify groups of similar agents is unable to introduce potential coalition members to each other. Clustering, on the other hand, is usually studied as a centralized problem. This paper reviews previous work on clustering and explores how a decentralized approach can be designed for a multi-agent system. The resulting procedure could be applied as a directory service and can enhance our understanding of coalition formation in general.

There are a large number of centrally controlled algorithms for discovering natural clusters, if they exist, in a data set (see [5] and [6] for a general review of the literature). There are three underlying methods relevant to this work; k-means, which divides points into k clusters centered around k chosen centroids, hierarchical clustering, which builds a series of clusterings

by repeatedly combining nearest neighbor clusters (or repeatedly splitting the weakest cluster), and density based clustering, which considers clusters to be connected areas of points with a density above a given minimum value. There are many advanced variations of these basic methods designed to optimize performance on particular types of data. These algorithms focus on finding clusters given various properties of the data set; clusters of widely differing sizes, odd cluster shapes, little separation between clusters, noise, outliers, high-dimensional data and complex data types for which a similarity function is difficult to define. In general, all clustering algorithms focus on creating good compact representations of clusters and appropriate distance functions between data points. To this purpose they generally need a user to provide one or two parameters that indicate the types of clusters expected. Most commonly, algorithms are given the number of clusters into which the data set is to be split, the size of desired clusters, or a density value that defines the expected distance between points within clusters. Since a central representation is available, where each point can be compared to each other point or cluster representation, points are never placed in a cluster with hugely differing members. Mistakes made by these algorithms instead take the form of incorrectly splitting a real data cluster in half or incorrectly combining two neighboring data clusters into a single cluster. On the whole, however, the definition of clustering is imprecise; the creation of clusters in which points have more in common with other cluster members than with members of other clusters. Given the complexities listed above it is usually not entirely clear what the “correct” clustering of a data set is. There is generally no one best algorithm for obtaining good clusters [4]. The most appropriate algorithm depends on the peculiarities of the data set considered.

This paper focuses on yet another complexity that must be faced in multi-agent systems; the distribution of the data over many machines. Part of the clustering process is to choose individual cluster characteristics, based on the particular composition of a data set, so as to best meet the given general guidelines. Thus, for instance, given a parameter of 10 for the desired number of clusters, actual cluster sizes are determined by the overall data set size and density variation. If data is decentralized, this process becomes much harder since information about the extent of the data set or even the density of points in a particular area is incomplete. Furthermore, in addition to determining cluster characteristics the actual points that best fit in a cluster also need to be found.

III. Agent Procedure

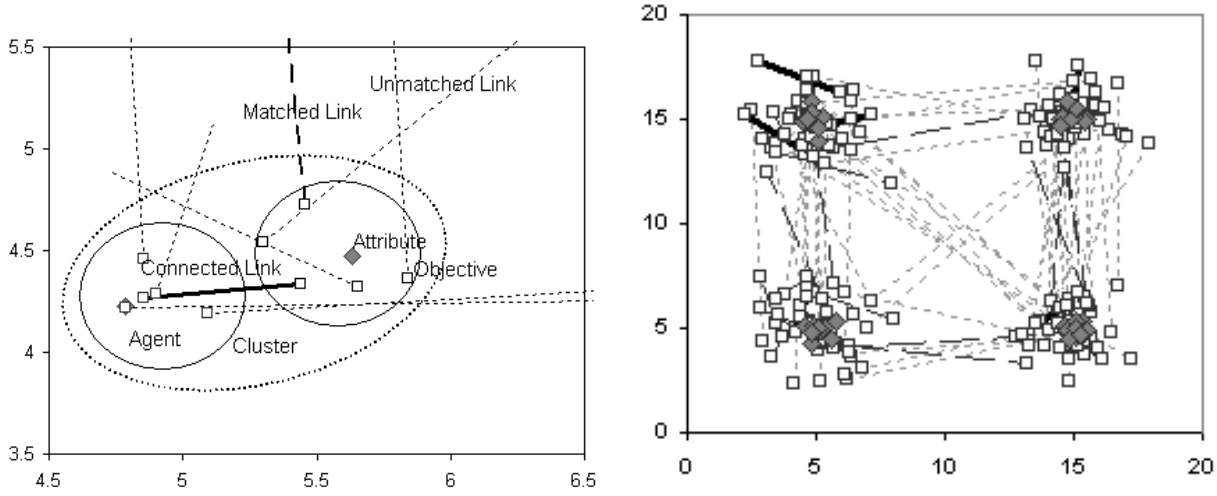
A. Model Description

In our model agents are defined by their characteristics, and can thus be seen, for the purpose of clustering, as a set of data items. Each agent has a small number of *links* to other agents. These links represent communication channels and thus define the neighborhood of an agent. The aim of the system is for agents to rearrange these links and to select some of them to form *connections* or *connected links* between themselves, generating a graph of connections corresponding to a clustering.

The creation of initial links is a bootstrapping problem; we assume they are derived from the placement of agents, or some other application-dependent source, and model them as a random network. Thus, in our simulations each agent starts out as a cluster of a single item with links to some other randomly chosen agents. As a simulation progresses, agents pick some of their links to become *matches*, or *matched links* based on the similarity of the agents joined by the link. Clusters choose the best of these matches proposed by their agents to become connections. Agents joined by a path of connected links form a single cluster.

The creation of connected links allows clusters to expand, but the initial clusters formed in this way are very poor. They represent the best clusters agents can see in an extremely limited local view. We give agents two behaviors that are used to improve clusters. First, agents in a cluster combine their individual pools of links, widening their individual neighborhoods. This allows them (still individually) to pick better matched links as candidates for cluster membership. Additionally, to prevent agents conglomerating into one large cluster, a limit is placed on cluster size. A further procedure then allows clusters to break weaker connections, enabling them to upgrade stronger available matches into connections. Since connections are between agents, breaking a connection can split a cluster, but leaves other stronger agent pairs connected in the resulting clusters.

We represent the search for good matches between similar agents as a matchmaking problem among agents' short-term objectives. Internally each agent is considered to have a main *attribute* that describes its basic characteristics. We would like to cluster agents according to these attributes. Each agent further contains a number of *objectives*, or current goals based on its attribute. In the first experiment in this paper an attribute is abstracted as a two-dimensional



(a) Two clustered agents.

(b) A small system.

Fig. 1. Diagrams of model components.

point. Objectives are represented by nearby points, chosen as a function of their agent's attribute. Objectives thus form a cloud of points around an agent's central data. For two-dimensional points objectives simply extend an agent's range of influence. For higher dimensional data, on the other hand, objectives can be chosen to reduce dimensionality and thus reducing the cost of checking for matches. In clustering text we use a full document as an agent's attribute and weighted word vectors to define its objectives. For more advanced agents, an objective could be only one of many tasks that an agent needs to complete to reach a final goal which is its central attribute. Figure 1(a) shows a diagram of two clustered agents. Figure 1(b) shows the links between all agents in a small system containing four clusters of ten points each.

Note that agents pick matches, since they are best able to determine how closely related their objectives are to other agents' objectives. Clusters, however, have a wider view of relative closeness on the attribute level since they contain a larger number of matched and connected links. Therefore clusters have a stronger basis on which to choose connections to make and break.

Multi-agent systems are often described in terms of their agents' individual behaviors. Along these lines, we can summarize the above agents as having the goal of finding good matches for their objectives. We would like the individual goals of these agents, along with the local coordination provided by cooperation within the clusters, to result in the overall system self-

organizing into a grouping of agents based on their attributes. In this paper we focus on the nature of this collective clustering behavior. For this reason agents are limited to having very simple decision processes. This allows us to highlight the basic clustering behavior and gives us a good foundation on which to study more complex agents that might be used in real applications.

B. Simulation Definition

Our aim is to examine the ability of a multi-agent system to find clusters in a set of points $P = \{x_1, \ldots, x_N\}$. A set of N agents $A = \{a_1, \ldots, a_N\}$ is created. Each agent, a_i, has as its attribute the point x_i in P. The agents are joined by links, forming a graph $G = (V, E)$ where the nodes are agents and the edges are links. We stipulate that each node in the graph G has the same degree δ. The interaction of the agents will change the edges in G, and eventually yield a new graph $G^* = (V, E^*)$, where E^* contains a set of chosen connected links. Connected components in G^* will correspond to clusters of P.

The procedure is as follows. Each agent, a_i, is given a set W_i of δ objectives, which are points (not necessarily in P) chosen as a function of x_i. Note that in the experiments presented in this paper we use $\delta = 5$ and all objectives of an agent a_i are given a common view of the agent's attribute, x_i. To initiate the system we chose for each objective $\omega_i \in W_i$ an objective $\omega_j \in W_j$ of a different agent uniformly at random in such a way that no objective is paired twice. This pairing of objectives leads to an initial set of *unmatched links*, denoted as E_0^-. The initial set of *matched links*, denoted as E_0^+, is set equal to the empty set. The initial set of *connected links*, E_0^*, is also empty, indicating that to begin with each agent forms a *cluster* of size 1. From this position we proceed in turns, each turn t consisting of the following four steps. Some of these steps contain functions, which will be defined later.

Step 1 **(Connecting):** Clusters choose some of their matched links from E_t^+ to become connected links using a rule r_c. Together with all links from E_t^* this forms the edge set E_{t+1}^*. Note that a connected link remains in E_t^+.

Step 2 **(Mixing):** Each cluster C_i has a set of unmatched links adjacent to it, given by

$$E_t^-(C_i) = \{(\omega, \omega') \in E_t^- : \omega \text{ is an objective of an agent in } C_i\}.$$

Each cluster mixes its objectives that are adjacent to an unmatched link in $E_t^-(C_i)$, using a random permutation. After each cluster has completed this mixing procedure,

a new set of unmatched links is obtained which is denoted by E^-_{t+1}.

Step 3 **(Matching):** All agents test their unmatched links (from E^-_{t+1}) using a turn-dependent matching probability $p^+_t(\omega,\omega')$. More precisely, an unmatched link (ω,ω') will become a matched link with probability $p^+_t(\omega,\omega')$ and will remain unmatched with probability $1-p^+_t(\omega,\omega')$. The new matched links together with E^+_t form E^+_{t+1}, and are taken out of E^-_{t+1}.

Step 4 **(Breaking):** Clusters choose some of their matched links from E^+_{t+1} to be broken, downgrading them to unmatched links and adding them to E^-_{t+1}, according to a breaking probability p_b. Each broken link is then removed from the set E^+_{t+1}. If a link to be broken is also a connected link, it is also taken out of E^*_{t+1}.

The connecting, mixing and breaking steps must be done collaboratively by a cluster as a whole, while the matching step can be done separately by each of a cluster's agents. In our simulations, to simplify operations within in a cluster, we select, at random, one cluster agent to perform the collaborative steps. Which agent is chosen is unimportant in these experiments since all agents in the simulation have equal capabilities. Over many turns the mixing and matching steps above create a search for matches among the objectives of neighboring clusters. The connecting and breaking steps result in clusters forming and changing over time.

To determine the matching probability, $p^+_t(\omega,\omega')$, each agent maintains a *range*, which it continuously adjusts as follows. Let R^i_t denote the range of a_i for turn t. The agent a_i considers M distances between objectives $\omega \in W_i$ and $\omega' \in W_j$ presented to it in the matching step. This might take several turns. After the M distances have been observed let σ be the smallest observed value. If $R^i_t \geq \sigma$, the agent forgets its distances and starts collecting M new distances. Meanwhile R^i_t stays the same. On the other hand, if $R^i_t < \sigma$, the range is gradually increased in the next M turns by a fixed fraction $(\sigma - R^i_t)/M$. However, if after say m turns, the agent is presented with a distance σ' smaller than the current σ, it repeats the test $R^t_i \geq \sigma'$ and follows the above procedure from that point on. The above procedure increases the range of the agent a_i. To decrease it, when a match is made the range is set to the distance of this match. In our experiments, $M = 100$.

For each turn t, we now let $p^+_t(\omega,\omega') = 1 - d(\omega,\omega')/R_t$. Here, $d(\omega,\omega')$ denotes some measure of the distance between ω and ω'. For instance, for spatial data we use the Euclidean distance between ω and ω'. In this paper we set the joint range for a link as the maximum of its two

objective's ranges, $R_t = \max\{R_t^i, R_t^j\}$, if $\omega \in W_i$ and $\omega' \in W_j$.

The rule used to choose connections, r_c, is defined as follows. Let the *strength* of a matched link (ω, ω') be defined as $1/d(\omega, \omega')$, meaning that more closely related links have a higher strength. All current matched links in the cluster, that are not connected, are first ordered according to their strength. We then proceed to create connections, starting with the strongest. A connection is created if the resulting cluster is not larger than a size limit L. Once a connection cannot be formed because of this size limit no more connections are formed. In the experiments in this paper L is set at 1.5 times the known cluster size.

To define the breaking probability we need a speed parameter λ. Consider a cluster C consisting of N_C agents. Each turn the cluster has a probability of breaking *one* of its links given as $p_b(C) = \lambda N_C/L$. The experiments in this paper use $\lambda = 0.3$, a value found to result in a good tradeoff between clustering speed and quality in [8]. The cluster chooses which link to break out of its set of matched links, $E^+(C)$, according to the following formula. Let $s(l)$ denote the strength of the link l, and let $s^C_{\max}$ be the maximal strength of a matched link in C. The weight of a link is defined as:

$$w(l) = (\frac{1}{s(l)} - \frac{1}{s^C_{\max}})^2.$$

The probability of the cluster choosing a link l is then given by:

$$p_b(l) = \frac{w(l)}{\sum_{l' \in E^+(C)} w(l')}.$$

IV. Experimental Results

This section presents the results of two experiments designed to measure the quality of clusterings produced by the agent method described in section III. The first experiment considers spatial data, for which there are a number of standard quality measures, and compares agent clusterings to those found by the k-means algorithm. The second experiment demonstrates how agents can be adapted for use with text data and gives some comparisons to the classical word vector model approach. To allow us to clearly separate the issue of decentralization from that of hard to distinguish clusters, we consider basic data sets in which clusters are clearly separable and of equal size.

A. Comparison to K-means Clustering

In order to precisely measure clustering ability we first examine the clustering of straightforward data sets made up of two dimensional spatial data points. As these data sets do not contain any of the difficulties addressed by more complex algorithms, the clusters found by the multi-agent system are compared to those found by Forgy k-means clustering, as described in [4]. K-means clustering works by choosing at random k initial cluster centers, or centroids, and assigning each data point to the cluster of the centroid to which it is closest. Centroids are then reset to be the center of the resulting clusters and the data points are reassigned to the new centroids. This process is repeated until a quality measure, the total squared error, stops changing. This is the simplest of the centralized clustering techniques. Nonetheless, it works well on the elementary data sets examined and illustrates the basic abilities and common mistakes of centralized clustering.

For this experiment each agent is given a two-dimensional spatial point as its attribute data, and agent objectives are simply also set to this data point. The strength of links is determined using the Euclidean distance between objectives as the measure of distance, $d(\omega, \omega')$. Table I compares the results of the agent clustering method to perfect, random and k-means clusterings for four data sets of varying sizes. The agent algorithm with run for 5000 turns with 50 trials for each data set. The k-means and random clusterings were run 100 times for each data set.

The data sets used were generated according to the procedure described in [13]. Each data set consists of K clusters of two-dimensional data points. A cluster is characterized by the number of points per cluster ($n_{\text{low}} = n_{\text{high}} = 100$) and the cluster radius ($r_{\text{low}} = r_{\text{high}} = \sqrt{2}$). The *grid* pattern is used, which places the cluster centers on a $\sqrt{K} \times \sqrt{K}$ grid. The distance between the clusters is controlled by k_{g}, which is set to 8. The noise parameter is set to 0. This creates a grid of well separated, circular, 2D clusters with 100 points each and equal density. Four data sets are generated with 25, 100, 400 and 1600 clusters. A corner of the 20×20 data set is shown in Figure 2.

We compare the quality of clusters found by our method to the generated clusters (perfect case), a set of clusters of the correct size but with randomly assigned points (random case) and the clusters generated by the Forgy k-means algorithm, given the correct value of k, initial centers picked uniformly at random from all the points, and run until no further improvement

	number of clusters			E^2			$\langle E^2 \rangle$	Rand	points out of place			
# points	min	avg	max	min	avg	max	avg	avg	min	avg	max	% total avg
PERFECT CLUSTERINGS												
2,500		25			5.04×10^3		2.02	1		0		0%
10,000		100			1.97×10^4		1.97	1		0		0%
40,000		400			7.94×10^4		1.98	1		0		0%
160,000		1600			3.16×10^5		1.97	1		0		0%
RANDOM CLUSTERINGS												
2,500	25	25	25	1.27×10^6	1.27×10^6	1.28×10^X06	509.64	0.924	2272	2289.6	2304	91.58%
10,000	100	100	100	2.09×10^7	2.09×10^7	2.09×10^X07	2090.38	0.980	9560	9576.4	9590	95.76%
40,000	400	400	400	3.37×10^8	3.37×10^8	3.38×10^X08	8429.88	0.995	38920	38943.9	38978	97.36%
160,000	1600	1600	1600	5.40×10^9	5.40×10^9	5.41×10^9	33773.95	0.999	156744	156772.6	156811	97.98%
KMEANS: k=100												
2,500	23	24.9	25	1.75×10^4	2.91×10^4	4.39×10^4	**11.65**	0.985	57	146.3	235	**5.85%**
10,000	97	99.2	100	8.86×10^4	1.20×10^5	1.62×10^4	**11.99**	0.996	368	576.9	857	**5.77%**
40,000	392	396.8	400	4.12×10^5	4.86×10^5	5.65×10^5	**12.14**	0.999	1939	2327.3	2730	**5.82%**
160,000	1580	1588.5	1598	1.78×10^6	1.92×10^6	2.08×10^6	**12.02**	1.000	31369	34398.6	37667	**21.50%**
AGENTS: $L = 150$, λ =0.3												
2,500	25	25.0	26	5.04×10^3	5.20×10^3	5.55×10^3	**2.08**	1.000	0	2.0	7	**0.08%**
10,000	100	100.0	101	2.02×10^4	2.12×10^4	2.30×10^4	**2.12**	1.000	6	12.4	24	**0.12%**
40,000	399	400.3	407	8.80×10^4	9.87×10^4	1.73×10^5	**2.47**	1.000	46	104.1	216	**0.26%**
160,000	1610	1625.6	1647	1.06×10^6	1.17×10^6	1.37×10^6	**7.33**	1.000	6320	7123.9	7996	**4.45%**

TABLE I

COMPARISON TO K-MEANS CLUSTERING FOR 2D SPATIAL DATA.

in clustering is found.

Several measures of cluster quality are compared. First, the total squared error metric, E^2, which is used by the k-means algorithm. Given k clusters $C_1, \ldots, C_k$, where C_i has a mean value m_i for $1 \leq i \leq k$,

$$E^2 = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - m_i||^2.$$

Total squared error gives an easily computed measure of the compactness of clusters, but in doing so favors small clusters. In fact, clusters with a single point have a square error of zero and thus the total squared error alone cannot be used to compare clusterings of a data set with different numbers of clusters. Total squared error also cannot be used to compare data sets of different sizes. The more points there are in a data set and the larger the range over which the data set is spread, the larger the total squared error.

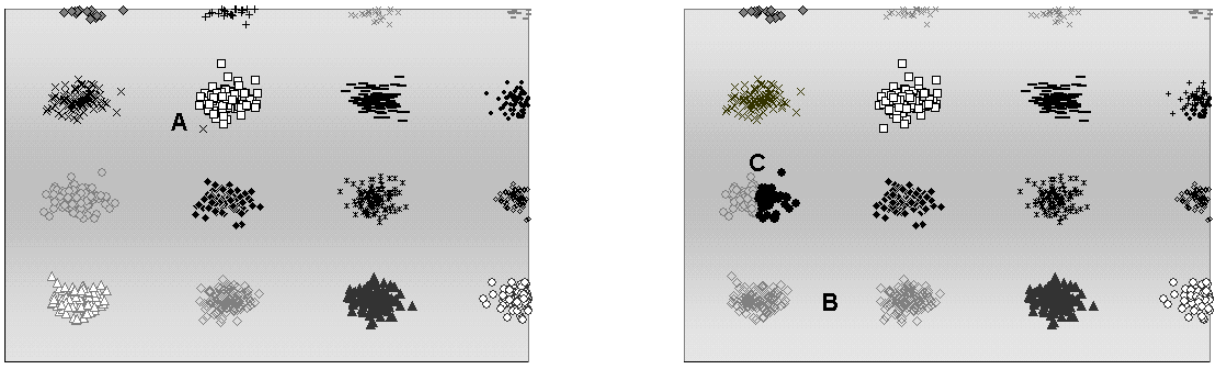
Jain and Dubbes [4] describe measures for comparing two clusterings of the same data by creating a contingency matrix which lists the number of points in common between each pair of clusters, between the two clusterings. These measures consider cluster membership, rather than the distance over which clusters are spread. We use this method to compare our experimental clusterings to the perfect clusterings produced by the generator. The Rand statistic [4] sums the

number of pairs of points that are correctly placed in the same cluster and the number of pairs of points that are correctly placed in different clusters and normalizes by the total number of possible pairs. However, for our data sets which have many small clusters, the number of pairs correctly placed in different clusters dominates. Thus we also use the contingency matrix to calculate the number of points incorrectly placed by associating with each real cluster the found cluster with which it has the most points in common. We sum the number of common points over all real clusters and subtract from the total number of points to get the points that are out of place. This gives a clearer distinction between clusterings that are close to, but not quite, correct. On the other hand it does not distinguish clusters that are incorrectly grouped into a single cluster. It also can count up to half of the points in a real cluster as incorrectly placed if that cluster is simply split in two.

In Table I it can be seen that while the agent-based clustering method usually does not find the perfect clusterings, it consistently improves upon the clusters found by the k-means algorithm. For smaller systems it places more than 99% of the points correctly and for the larger system it places at least 95% of points correctly. The reason why a less good clustering is found for the largest data set is the amount of time it takes the agent system to find clusters. During a trial the agents rapidly find a set of mediocre clusters and then improve these over time. In this experiment the agent systems were run for 5000 turns, which is a long time for the smaller systems, but halts the largest system while it is still improving. In [8] we present detailed results on timing.

B. Nature of Discovered Clusters

Figure 2(a) shows the clusters found by the agent procedure in a sample run, for a section of the 10×10 cluster grid. Figure 2(b) shows, for comparison, the clusters found by an example k-means run. Both methods were given the correct input parameters. The agent-based procedure used a maximum cluster size, L, of 150, or 1.5 times the cluster size. The k-means algorithm was run with $k = 100$, the number of clusters in the data set. Overall both methods found the correct number of clusters, meaning that k-means did not loose any, which is possible, and that our agents adjusted to the correct cluster size of 100 instead of staying at the maximum size of 150. Total squared error for the k-means run was 126,889 versus 20,372 for the agent-based method.



(a) Clusters produced by the agent-based procedure.

(b) Clusters produced by the k-means algorithm.

Fig. 2. Agent-based and *k*-means clustering results.

The graphs in Figure 2 show the typical mistakes made by each method. At point A the agent method has associated a point with a neighboring cluster instead of with the correct cluster. This can happen when a point has joined the correct cluster, but has been broken off again due to the randomness of the breaking function. Agent clusterings remain dynamic, and generally these points reattach, though they can spend some time as a member of a neighboring cluster. Thus at any point in time after good clusters have been found, several such mistakes are likely to exist. This type of mistake can also occur when a point simply does not find its correct cluster. This results in clusters with one or a small number of agents that can take a long time to discover their correct group. We see in Figure 2a that the k-means algorithm, by contrast, makes very different types of mistakes. At label B it has incorrectly joined two clusters into one, and at label C it has incorrectly split a cluster in two equally sized components. This can occur when two initial centroids are chosen from the same cluster. This cluster then becomes split, but somewhere else two clusters need to be joined to maintain *k*. When there are large numbers of clusters it is likely that this will occur. There are heuristic methods of choosing better initial centriods, however a perfect choice would amount to knowing the correct clustering a-priori.

C. Clustering Text

Text clustering uses the same basic methods as spatial data clustering, however it faces the additional difficulty that the similarity of text documents is ambiguous. Since there is no strict distance measurement that can be used, text algorithms focus on effective ways to extract a document's meaning through identifying content bearing words in the text or by means of any

weighting function	similarity coefficient	number of clusters			points out of place		
		min	avg	max	min	avg	max
none	Dice	11	15.46	25	4	10.76	27
≈IDF	$\sum_{k=1}^{m} \min(t_{ik}, t_{jk})$	10	13.5	19	5	11.1	19
≈IDF	$\sum_{k=1}^{m} s(k)$, where $s(k) = t_{ik} + t_{jk}$ if $t_{ik} > 0$ and $t_{jk} > 0$, and $s(k) = 0$ otherwise.	10	13.5	19	0	6.58	17

TABLE II

RESULTS OF TEXT CLUSTERING AGENTS: 50, 500 TURN, RUNS PER EXPERIMENT.

additional information available through user feedback or references [2]. Again, in this paper we wish to keep the issue of decentralization separate from that of difficult to find clusters. Thus we consider a straightforward data set and use comparison methods from the classical word vector model [10].

We use a 100 point data set containing the first ten chapters of each of ten books chosen to have distinct subjects. These chapters are between 343 and 15733 words long. Agents are each given the entire text of a chapter as their data point. Following the word vector model agents process these texts to create a weighted vector of all unique words (without stopping or stemming). We represent this weight vector as $T = \{t_1, \ldots, t_m\}$, where there are m unique terms in the entire document set, though of course agents only know of the terms their text contains and thus have a 0 weight for all other terms. Table II shows data for agent clusters found using three different combinations of weighting schemes and distance functions. In the first of these all words in the text are given a weight of 1 and we use the Dice coefficient:

$$D(T_i, T_j) = \frac{2(\sum_{k=1}^{m} t_{ik} \cdot t_{jk})}{\sum_{k=1}^{m} t_{ik} + \sum_{k=1}^{m} t_{jk}}$$

This gives a measure of the number of words two texts have in common, normalized by the lengths of the texts. We see in Table II that even with this simple method agents are already able to find passable clusterings.

A more advanced method of weighting terms is to use each word's inverse document frequency:

$$IDF(t_k) = \log\left(\frac{N}{n_k}\right) + 1$$

where N is the total number of documents, and n_k is the number of documents in which term t_k appears. This weighting method is based on the assumption that words that occur very frequently or almost never among the documents are likely to be functional words, while those with a middling frequency are likely to be content bearing. We see here a problem, however, in

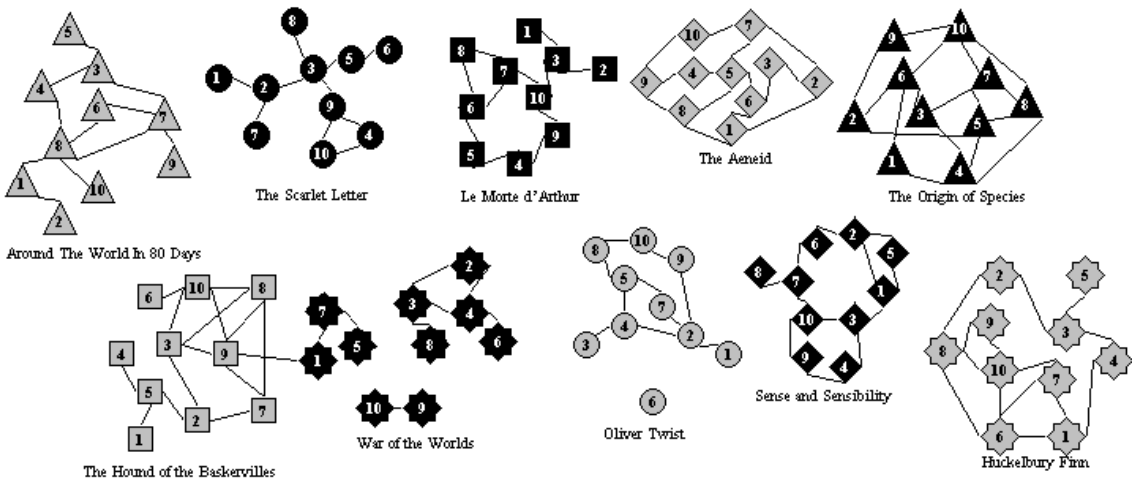


Fig. 3. Example clustering of text data set.

that in order to calculate the inverse document frequency of a word an agent must examine the entire document set. Since this is not possible in a decentralized setting we instead have agents attempt to estimate inverse document frequencies by observing the frequency of words in the neighbors with which they attempt to make matches:

$$\approx IDF(t_k) = \log\left(\frac{\text{total matches attempted}}{\text{potential neighbors that contained term } k}\right) + 1.$$

To account for difference in document lengths agents normalize their weights so that their entire vector sums to 1. Agents then use one of the simpler of the methods of determining similarity, the sum of the minimum of the weights of each term, though we find in the last row of Table II that using the sum of the combined weights for each common term works slightly better. Since changing the weights of words changes link strengths and thus requires updating cluster heads we have agents reevaluate weights only after each 1000 attempts to make a match.

Figure 3 shows an example average clustering using the last of the three methods in Table II. The clustering divides the data set into 12 clusters and places 6 points incorrectly. From the diagram we can see that this is actually quite a reasonable clustering of the data set. The mistakes that are made are similar to those seen for spatial data, the “War of the Worlds” cluster has been broken up, one of its components has joined “The Hound of the Baskervilles” and one of the “Oliver Twist” points has gotten lost. Considering that agent clusterings are dynamic these mistakes are most likely to be temporary, so that over time the average clustering will be the correct one.

	Around The World In 80 Days (2831 words total)				La Mort d'Arthur (979 words total)				Sense and Sensibility (2506 words total)			
agent rank	word	agent score	actual rank	actual score	word	agent score	actual rank	actual score	word	agent score	actual rank	actual score
1	fogg	0.034	1	33.03	unto	0.034	1	30.79	dashwood	0.037	1	33.03
2	phileas	0.034	2	33.03	knights	0.033	5	25.47	norland	0.032	3	27.88
3	bombay	0.028	3	25.47	uther	0.032	6	25.47	elinor	0.031	2	27.88
4	suez	0.028	5	25.47	merlin	0.032	3	27.88	marianne	0.028	7	25.47
5	passport	0.027	8	22.64	king	0.030	8	23.86	park	0.028	4	27.18
6	passepartout	0.026	4	25.47	arthur	0.030	2	28.78	handsome	0.028	5	26.51
7	p.m	0.025	7	22.64	brastias	0.030	4	25.47	marianne's	0.027	8	25.47
8	mongolia	0.022	17	19.98	ulfius	0.027	10	22.64	barton	0.026	11	22.64
9	club	0.022	6	23.12	wherefore	0.026	7	24.64	john	0.025	6	26.09
10	steamers	0.022	18	19.98	barons	0.025	9	22.64	mrs	0.024	9	23.86
11	steamer	0.022	9	22.64	lords	0.023	16	17.42	middleton	0.024	12	22.64
12	october	0.022	12	20.91	archbishop	0.023	12	19.98	income	0.024	20	19.98
13	repaired	0.021	20	18.86	kay	0.023	14	19.98	wishes	0.023	14	22.33
14	visaed	0.021	31	16.88	ye	0.022	11	21.48	daughters	0.022	10	23.57
15	departure	0.020	23	18.20	tintagil	0.022	24	16.88	dashwood's	0.022	19	19.98
16	eighty	0.020	16	19.98	igraine	0.022	13	19.98	propriety	0.021	21	19.98
17	robber	0.020	15	20.39	wales	0.020	48	13.49	situation	0.021	13	22.33
18	reform	0.020	10	21.83	pentecost	0.020	44	13.49	conversation	0.020	23	19.54
19	detectives	0.020	38	15.98	ector	0.019	21	16.88	edward	0.020	26	18.86
20	brindisi	0.020	28	16.88	purvey	0.019	23	16.88	cottage	0.019	15	21.56
21	paris	0.019	21	18.48	merlin's	0.019	41	13.49	affection	0.019	17	20.62
22	consulate	0.019	72	13.49	canterbury	0.018	18	16.88	herself	0.019	16	20.99
23	quay	0.019	80	13.49	pendragon	0.018	22	16.88	dashwoods	0.018	32	16.88
24	detective	0.019	24	17.42	nourish	0.018	43	13.49	elinor's	0.018	33	16.88
25	doesn't	0.019	39	15.98	assay	0.018	36	13.49	required	0.018	27	18.27

TABLE III

TOP 25 WORDS FROM THREE EXAMPLE CLUSTERS.

Finally Table III shows for three of the clusters in Figure 3 the top twenty-five words, scored by adding the weights of all agents in the cluster that contain that word. In addition, the table shows the scores and ranks these words would have if agents used the actual inverse document frequency of the words for this data set as their weights. We see from this table that the agents were able to find reasonably good estimates of the central inverse document frequency values.

V. CONCLUSION AND FUTURE WORK

The experiments reported in this paper indicate that decentralized agent systems can indeed be used to find clusterings of data sets, with surprisingly good quality. Our method contains two parameters, the maximum cluster size and the speed at which clusters break, that can be adjusted to trade off knowledge of the data set and time for cluster quality. Agent clusters show an ability to adjust their size to the size of underlying data clusters, and to learn the appropriate range for matching. In addition, more complex text agents show an ability to learn reasonable approximations of word frequency within a data set. In other work we have shown that agents

can learn a much wider range of appropriate cluster sizes by watching the series of links they make and break [9] and have found that agent clustering show good scalability properties [8]. Together these results demonstrate how using rational autonomous agents, which can modify their decision making criteria over time, can be advantageous in the clustering problem.

The agents studied in this paper are extremely simple and clustered straightforward data sets. However, they exhibit the basic dynamics of forming clusters based on similarities between agent attributes. The text agents in this paper demonstrate how these agents can be modified to deal with more complex forms of data and matching functions. They further suggest that approximate values of central concepts like word frequency might be sufficient to produce good clusterings. On the other hand, the text data set studied was fairly small and this work needs to be extended for larger data sets. Further research directions include studying more advanced matching functions and distance measures in order to tackle issues, such as clusters of varying size, shape and density, addressed by more advanced central clustering algorithms. How well this works in practice must be determined by future research, however, some such form of decentralized agent grouping may in the future provide a basis for peer-to-peer directory services.

REFERENCES

[1] Decker, K., Sycara, K., and Williamson, M.: Middle-Agents for the Internet. Proceedings of the 15th International Joint Conference on Artificial Intelligence (1997). 578-583

[2] Faloutsos, C., and and Oard, D.: A Survey of Information Retrieval and Filtering Methods. Technical Report CS-TR3514, Computer Science Department, University of Maryland, 1995.

[3] Forner, L.: Yenta: A Multi-Agent, Referral-Based Matchmaking System. The First International Conference on Autonomous Agents (1997) 301-307

[4] Jain, A., and Dubes, R.: Algorithms for Clustering Data. Pentice-Hall advanced reference series. Prentice-Hall, (1988)

[5] Jain, A., Murty M., Flynn, P.: Data Clustering: A Review. ACM Computing Surveys, Vol 31, No. 3, September (1999). 264-322

[6] Kaufman, L. and Rousseeuw, P.: Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley and Sons (1990)

[7] Klusch, M. and Gerber, A.: Dynamic Coalition Formation Among Rational Agents. IEEE Intelligent Systems. Vol 17, No. 3 (2002) 42-47

[8] Ogston E., Overeinder, B., Van Steen, M., and Brazier, B. A Method for Decentralized Clustering in Large Multi-Agent Systems. Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (2003) 789-796.

[9] Ogston E., Overeinder, B., Van Steen, M., and Brazier, B. Group Formation Among Peer-to-Peer Agents: Learning Group Characteristics. Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC) (2003) to be published (Springer - Lecture Notes in Computer Science series vol. no. 2872)

[10] Salton, G. and McGill, M.: Introduction to Modern Information Retrieval, McGray-Hill, 1983.

[11] Sandholm, T. and Lesser, V.: Coalition Formation among Bounded Rational Agents. 14th International Joint Conference on Articial Intelligence (1995) 662-669

[12] Shehory, O. and Kraus, S.: Task Allocation via Coalition Formation among Autonomous Agents. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (1995) 655-661

[13] Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH:A New Data Clustering Algorithm and Its Applications. Data Mining and Knowledge Discovery, 1(2) (1997) 141–182