Available online at www.sciencedirect.com
SciVerse ScienceDirect
Procedia Computer Science 19 (2013) 503 – 510
Procedia Computer Science

The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)

Unsupervised Mining of Activities for Smart Home Prediction

Jérémy Lapalu*[a], Kevin Bouchard[a], Abdenour Bouzouane[a], Bruno Bouchard[a], Sylvain Giroux[b]

[a] *LIARA Laboratory, Université du Québec à Chicoutimi (UQAC) Chicoutimi, G7H 2B1, Canada*
[b]*DOMUS Laboratory, Université de Sherbrooke, J1K2R1, Canada*

Abstract

This paper addresses the problem of learning the Activities of Daily Living (ADLs) in smart home for cognitive assistance to an occupant suffering from some type of dementia, such as Alzheimer's disease. We present an extension of the Flocking algorithm for ADL clustering analysis. The Flocking based algorithm does not require an initial number of clusters, unlike other partition algorithms such as K-means. This approach allows us to learn ADL models automatically (without human supervision) to carry out activity recognition. By simulating a set of real case scenarios, an implementation of this model was tested in our smart home laboratory, the LIARA.

Keywords : Flocking; Data mining; Clustering; Smart home;

1. Introduction

Due to the aging of the population and its consequences, the number of people suffering from category of disorders known clinically as dementias, like Alzheimer's disease, grow up every year [1]. In this context, many researchers [2, 3] try to develop smart home technologies to provide cognitive assistance to a resident's in his everyday Activities of Daily Living (ADL). The identification of the ongoing inhabitant ADL is one of the main issues in smart homes. In most cases, the activity is inferred with a plan library that consists of basic activities predefined by an expert [4, 5]. Due to the high number of activities and their complexity, the resulting library is usually incomplete. Another option is to build it automatically thanks to a supervised learning method. Still, it requires interventions from a human to label the dataset. Such constraint is possible on a small set of data, but prevents the technique from being exploited in real-

* Corresponding author. Tel.: +1-418-545-5011 extension 5604
E-mail address: jeremy.lapalu@uqac.ca.

life contexts since it is very difficult to enumerate all the possible activities. Nevertheless, new ways allow us to explore this paradigm with unsupervised learning methods, which eliminate the requirement of a human in the process.

To do so, one avenue is to exploit clustering algorithms to learn patient's activities automatically such as the well-known K-means algorithm and its variants [6, 7]. However, K-means needs an initial number of clusters for its execution. This limit its applicability in our context, since it is hard to know the number of possible activities other than by trial and errors. In recent years, new clustering algorithms [8, 9] were developed on an emergent behavior called Flocking [10]. Flocking reproduce a behavior rising from the group movement of several societies in nature such as flocks of birds, schools of fish, and colonies of bees. In this method, an entity is called a boid. Each boid makes its own decisions on its movement according to simple rules. It reacts to other characters perceived within its local neighborhood. The Flocking model does not require an initial number of clusters unlike other partition clustering algorithms. We propose, in this paper, an activity clustering algorithm based on the Flocking, which is specifically adapted to unsupervised learning in smart homes. Our contributions are twofold. First, we provide a new experiment to the field of data mining on the Flocking algorithm to address the issue of fast evolving data [11]. Second, we are the first to exploit a Flocking based method to perform activity recognition in smart home while most existing approaches are supervised [12]. Moreover, only few completely unsupervised methods exist [13] to address this issue, and they are very limited. For instance, they do not take into account the fast evolution of the data in our context and they are based on key objet relevance which limit them to a small set of possible ADLs [14].

This paper is organized as follows. The next section draws an overall picture of the existing work on clustering algorithms and data mining methods in smart home. The third section presents our extension of the Flocking. The fourth section shows how this model is implemented in the LIARA smart home laboratory. The fifth section presents the results of our experimental phase using a set of real case scenarios. Finally, the last section presents our conclusion and future work.

2. Related Works

Clustering involves separating a large set of data into subsets named clusters. A cluster is a collection of data; elements within a cluster are similar between them, and are dissimilar with other clusters' elements [15]. The goal behind clustering is to find inherent structure in the data and to show this structure as a set of clusters [16]. Two main clustering techniques exist: partitioning and hierarchical [15]. Hierarchical methods build a hierarchy of clusters and a unique partition of objects. In these techniques, the number of clusters to construct does not need to be specified and a distance matrix is usually used as clustering criteria. Although these methods are often portrayed as a clustering approach with better quality, in the general case, the complexity of hierarchical clustering is at least $O(n^2)$, which makes them too slow for large data sets. On contrary, the time complexity of partitioning techniques is almost linear. These techniques break up data into a set of non-overlapping groups to maximize the evaluation value of clustering. The K-means is a partitioning algorithm; it is based on the sound foundation of analysis of variances. Each cluster starts with a random center. Then, the algorithm reassigns the data objects in the dataset the centers of the clusters based on the similarity between the center and the data object. This reassignment procedure stops when a convergence criterion is met or after a fixed number of iterations. The major drawbacks of the K-mean algorithm are that the clustering may converge to the local optima, and it is sensitive to the selection of the initial cluster centroids. It also requires a prior knowledge of the problem to select the number of clusters needed.

To address the limitations of classic partitioning methods, researchers in computer science have proposed several approaches such as Ant clustering [17, 18] and Flocking [8, 9]. Like Flocking, Ant clustering does not require a predefined number of clusters. Deneubourg and al. [18] presented a basic model to explain the behavior based on the group movement of corpses and eggs in real ant colony : An ant, named an agent and represented on a 2D grid, follows one simple rule: “randomly moving in the grid and establishing a probability of picking up the data object it meets if it is free of load or establishing a probability of dropping down the data object if it is loading the data object”. On the same way, the Flocking algorithm generates a clustering of a dataset on a 2D grid. Nevertheless, the Flocking is more efficient than Ant colony because each data is an agent in the virtual space, and each agent is moving according to a heuristic, unlike the random activity in Ant clustering.

3. The extension of the Flocking algorithm

The basic Flocking model consists of three simple steering rules: alignment, separation and cohesion [10]. These rules are executed each iteration by all individual agent. With these three rules, each agent gets closer to his neighbors indiscriminately adopting a herd pattern. Therefore, only one cluster is formed after several iterations. That’s why we implement two new rules to exploit it as a clustering algorithm: similarity and dissimilarity. These rules allow us to create many clusters because similar agents follow each others, and dissimilar agents tend to separate. In addition, if an agent finds himself alone in the base model of Flocking, it stops moving. To correct this problem, we slightly modified the rule of alignment so that the agent continues to move straightforward if it is left alone. Besides, the base complexity is $O(n^2)$, but by dividing the virtual world into zones of equal size we can reduce it to $O(n)$. Linear complexity is desirable due to the large amount of data that must be processed. In the next subsections, we describe the five rules that we proposed, with their mathematical formalization.

3.1. Alignment, Separation and Cohesion forces

Alignment force attempts to keep an agent’s heading aligned with its neighbors. The force is calculated by first iterating through all the neighbors and averaging their heading vectors. Considering that k is the total number of current agent’s local neighbors, $\overrightarrow{Ha}$ is the agent’s heading force, and $\overrightarrow{Hn}$ is a neighbor heading force, the definition of $\overrightarrow{F_A}$ the force driven by alignment rule is:

$$\overrightarrow{F_A} = \frac{1}{k+1}\left(\overrightarrow{Ha} + \textstyle\sum_n^k \overrightarrow{Hn}\right) \tag{1}$$

Separation force creates a force that steers an agent away from those in its neighborhood region. When applied to a number of agents, they will spread out, trying to maximize their distance from every other agent. Considering that k is the total number of current agent’s local neighbors, $\overrightarrow{Pa}$ is the current agent’s position vector, and $\overrightarrow{Pn}$ is a neighbor’s position, then $\overrightarrow{F_P}$ is the force driven by separation rule defined by the definition below:

$$\overrightarrow{F_P} = \textstyle\sum_n^k \frac{\overrightarrow{Pa}-\overrightarrow{Pn}}{\left\|\overrightarrow{Pa}-\overrightarrow{Pn}\right\|^2} \tag{2}$$

Cohesion force produces a steering force that moves an agent toward the center of mass of its neighbors. This force is used to keep a group of agents together. We calculate the average of the position vectors of the neighbors. This gives us the center of mass of the neighbors, the place the agent wants to get to, so it seeks to that position. The force $\overrightarrow{F_C}$ is the definition below where $\overrightarrow{Pa}$ is the agent’s position, Ms is the agent’s maximum speed (a predefined constant), $\overrightarrow{Va}$ is the agent’s velocity, and $\overrightarrow{CoM}$ is the center of mass of the boid. $\overrightarrow{CoM}$ is determined by k the total number of current agent’s local neighbors, and $\overrightarrow{Pn}$ is a neighbor’s position vector.

$$\overrightarrow{F_C} = \left(\frac{\overrightarrow{CoM}-\overrightarrow{Pa}}{\left\|\overrightarrow{CoM}\right\|} * Ms\right) - \overrightarrow{Va}\,, \qquad \overrightarrow{CoM} = \frac{1}{k}\left(\textstyle\sum_n^k \overrightarrow{Pn}\right) \tag{3}$$

3.2. The Flocking extension to clustering

Dissimilarity force creates a force that steers an agent away from those in its neighborhood like the separation rule, but only between dissimilar agents. The mathematical implementation of the dissimilarity force (see equation 4) extend the separation rule by modulating $\overrightarrow{\mathrm{Pa}} - \overrightarrow{\mathrm{Pn}}$ in function of $d^*(a,b)$ which represent the dissimilarity between a couple of agents a and b. The function $d^*(a,b)$ is the normalized Euclidian distance represented on (5) which compare agents' data in order to compute a certain distance between a and b. In the Euclidian equation, n denotes the dimension of the vector space related to the attributes of the learning data set.

$$\overrightarrow{F_D} = \sum_n^k \frac{(\overrightarrow{Pa}-\overrightarrow{Pn}) * d^*(a,b)}{\|\overrightarrow{Pa}-\overrightarrow{Pn}\|^2} \tag{4}$$

$$distance(a,b) = \sqrt{(x_1^a - x_1^b)^2 + \cdots + (x_n^a - x_n^b)^2} \tag{5}$$

Similarity force produces a steering force likewise to the cohesion rule, but only between similar agents. However, a new center of mass ($\overrightarrow{\mathrm{CoM_S}}$) calculation, described below, replace the one from equation three.

$$\overrightarrow{F_S} = \left(\frac{\overrightarrow{CoM_S}-\overrightarrow{Pa}}{\|\overrightarrow{CoM_S}\|} * Ms\right) - \overrightarrow{Va}, \qquad \overrightarrow{CoM_S} = \frac{1}{k}\left(\sum_n^k \left(\left(\overrightarrow{Pa} - \overrightarrow{Pn}\right) * S(a,b) * \overrightarrow{Pn}\right)\right) \tag{6}$$

Where k is the total number of current agent's local neighbors, Pa is the current agent's position, Pn is a neighbor's position, and $S(a,b)$ the similarity between agents a and b.

$$S(a,b) = 1 - d^*(a,b) \tag{7}$$

To achieve a complete Flocking behavior, the results of all rules are weighted and summed to give a steering force that will be used by the current agent for calculate his next velocity. If w_X are the predefined weight values, then equation 8 determine $\vec{F}$ our Flocking force which can be seen as a resulting force from the linear combination of all the other forces. If any of the force exceeds the maximum force, then it is not added.

$$\vec{F} = w_S\overrightarrow{F_S} + w_D\overrightarrow{F_D} + w_C\overrightarrow{F_C} + w_P\overrightarrow{F_P} + w_A\overrightarrow{F_A} \tag{8}$$

4. Smart home validation

The LIARA[1] lab consists of a standard apartment (kitchen, living room, bedroom, and bathroom) equipped with sensors, smart tags (RFID), pressure mats, localization/identification systems for objects and residents, audio and video devices, etc. Our research project aims to explore ways to provide pervasive cognitive assistance to people suffering from Alzheimer's disease. As we noted beforehand, the main issue in this project is to recognize and predict the inhabitant ADLs. In this context, we need to represent the data as a set of mobile boid, also called agent, to use the Flocking based algorithm. The values of the sensors, which are stored in a database, change frequently due to patient's actions. These modifications are called events, and we represent our agents with these events. When an event is detected, a new agent is created with the modified values of the corresponding sensor or RFID attributes. An agent is essentially defined by three main attributes: spatial region, position and the generation time of the event. After its creation, an agent applies immediately its Flocking behavior, described by the rules of section 3.

[1]Laboratoire d'Intelligence Ambiante pour la Reconnaissance d'Activités (http;//liara.uqac.ca)

4.1. The similarity and dissimilarity metrics

The definition of the similarity/dissimilarity of two agents is an important issue for a good clustering. A lot of parameters can be taken into account: event's duration, time between two events, event's type (moving object, immobile object), spatial relation, spatial region (bathroom, kitchen, bedroom, etc.), position, event's generation time (morning, afternoon, night, etc.). Among these variables, we chose to investigate three attributes: temporal relation between two events, spatial region and Cartesian positions. The comparison of two agents' data check if their parameters match with each other: small time between them, close position and same spatial region. Such agents are considered similar and are attracted to each other because of the similarity rule. On contrary, if their spatial regions are different, or they are far, or the time between them is great, the dissimilarity rule is applied and they move in opposite direction.

5. Experiments and results

To test the efficiency and accuracy of our new model, we have conducted extensive experiments in our smart home. A participant was asked to perform normally specific scenarios without precise indications. A description of the tests' dataset is given on Table 1.

Table 1 – The scenario dataset

Scenario name	*Number of events*
Cook for lunch	103
Cook for dinner	98
Go to the toilet	14
Read a book	12
Sleep in the bedroom	16

Events represent changes in the state of the different sensors of the environment. This is why simple activities like reading a book contain fewer events than tasks needing more complex operations (cooking). Indeed, fewer sensors change of states when a resident is only reading a book.

Each scenario has allowed to generate a text file that contains 6 data per row, and each row symbolizes an event. The Table 2 contains a part of the scenario cook for lunch. The columns X and Y are the position of sensors and RFID tags; they are ranged between 0 and 600. The time parameter is expressed in minutes from 0h00; 710 min is 11:50 AM for example.

Table 2 – Little part of cook for lunch scenario

Type	*Name*	*Location*	*X*	*Y*	*Time*
Sensor	TC2	Kitchen	250	250	710
Sensor	CA5	Kitchen	300	100	710
RFID	Pan	Kitchen	362	139	710
...	...	...	...	...	...

5.1. Experimental setup

For the experimental phase, we decided to run two other clustering algorithms with our new Flocking based model. Our choice fell on K-means because of its high efficacy. We also chose Expectation-

Maximization (EM) which is also a clustering algorithm (among other things) that can either be used with a fixed number of clusters or that can efficiently estimate it. For the three versions, we used the same dataset and the Euclidian distance measure. We used Weka to test EM and K-means, and we used the Java language to implement our Flocking algorithm. Each data is represented as one boid, and all boids follow the five rules mentioned in section 3. Each boid can only sense other boids located within its neighborhood distance. Higher is the sensing distance and faster the clustering result emerges. However, at the same time, each agent needs more computational time to generate its moving direction and speed at each iteration. The virtual grid space for agents is set as a 500 x 500 2D square space. Likewise to others data mining techniques, we separate the learning phase and the testing phase. In the learning phase 2/3 of data are used for creating the clusters, and 1/3 for testing these clusters.

The initial distribution of the experimental dataset is shown on Fig. 1 (a). As can be seen, at the beginning of the learning phase one unique cluster exists and contains all agents. We let the agents move freely according to the movement forces we defined in section 3 once per iteration (every 40ms). At the end of an iteration, clusters are built recursively from the agent list. If the square distance between two agents is under to their view distance, they are in the same cluster. When all agents have been assigned to a cluster, the center and the radius of the clusters are calculated relative to the agents inside. The Fig. 1 (b) shows that five clusters are created in the learning phase after several iterations. Each of them is relative to a scenario presented in dataset. This means that our algorithm succeeds in finding the exact number of activities during the learning phase, without supervision. After several iterations, if there are no more changes in the number of clusters, the testing phase begins.

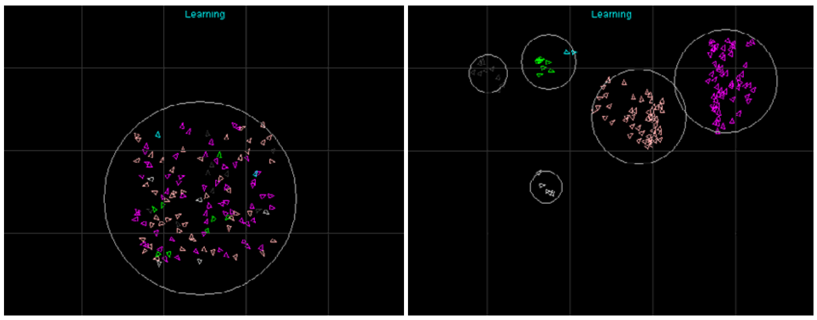

Fig. 1 – (a) Initial data distribution in the learning phase; (b) Result after 1000 iterations.

5.2. Testing phase

This phase aims to control the quality of our clusters. At the beginning of the testing phase, a representative agent is created for each cluster. It computes the average of the data of all agents in their clusters. After the creation of these representative agents, all others agents are destroyed. The 1/3 of the dataset unused replace these destroyed agents. In this phase, clusters are built from each representative agents created at the beginning. This allows to confirm if the clustering is efficient or not. The percent of success of each cluster is calculated every frame, and the average of all cluster is inferred. After 25 measures, an average of these success rates is saved to a log file. The success of a cluster is the number of agents of the same class divided by the total number of agents in the cluster.

5.3. Experimental results

We ran the algorithm 10 times with these 5 real case scenarios, and we computed the average of the results. The Table 3 shows a part of our results.

Table 3 – The average results of Flocking clustering Algorithm on real datasets

Iterations	3371	4591	5753	6897	8064	8882	10227	13404
Success	0.6359	0.7184	0.8038	0.8716	0.8863	0.8902	0.9000	0.9249

An iteration is the calculation of the combined forces $\vec{F}$ for all agents, and the update their position. The success rate is expressed between 0 and 1. The closer the value is to 1, the purer the cluster is. As one can see, the success rate rises rapidly at the beginning across the iterations to around eighty-seven percent and then progressively to reach the threshold of ninety-two percent after approximately thirteen thousand iterations. This means that for the five activities that the algorithm detected during the learning phase, the average purity of clusters is around ninety-two percent. The success rate is not equal to one hundred percent due to noise (radio-frequency interference, sensors failure, etc.), and this is to be expected due to unrelated events generated from the sensors imprecision. To compare, the Table 4 presents the results obtained with K-means and EM (Expectation Maximization) algorithms.

Table 4 – The results of K-means and EM algorithms on real datasets

Algorithm	*Iterations*	*Success*	*Number of clusters*
K-Means	5	0.6033	5 (set at start)
EM (k set)	5	0.7603	5 (set at start)
EM (k unset)	5	0. 6240	7

These algorithms are more efficient in terms of iterations number. However, they cannot achieve a high success rate partly because of the small number of data contained in our dataset. Moreover, K-means requires to know the number of clusters at the start. Furthermore, EM cannot find the exact number of activities, and its success rate is lower if we don't set the number of clusters. Another important thing to note is the difference in the way the iteration metric is calculated. While an iteration in classic data mining approaches represents the complete reattribution of the elements to cluster, in our Flocking extension it is only a small movement for each boid. Therefore, the difference in performance of our algorithm is not as big as it seems. Besides, as stated in the beginning, the Flocking computational complexity is linear as for the partitioning algorithms with a fixed k and as opposed to hierarchical methods, which are polynomial.

6. Conclusion

In this paper, we have presented a new algorithm of clustering based on the Flocking as a potential solution to the problem of mobile data. The emergent behavior of Flocking combined with the addition of two more rules (similarity and dissimilarity) allows us to infer the exact number of ongoing activities even with few data. Moreover, in our context of activity recognition, we need an unsupervised method to recognize the ADL of the patient observed. The Flocking does not require a predefined number of clusters at start unlike others classic clustering algorithm like K-means. This is essential because we do not know in advance the number of activities a resident will perform in its daily life. While classic clustering methods possess their own advantages over our method, in our context we demonstrated the Flocking extension was more suited and performed better than K-means and EM. In future work, we aim to extend this approach in order to introduce more types of criterion such as advanced spatial or temporal reasoning. Moreover, we plan on doing large-scale experiments on data collected over a long period of time.

Acknowledgements

We would like to thank our main financial sponsors: the Natural Sciences and Engineering Research Council of Canada, the Quebec Research Fund on Nature and Technologies, the Canadian Foundation for Innovation. We would like to thank our health regional center for providing us the Alzheimer participants. Finally, special thanks to our neuropsychologist partner and her graduate students who indirectly worked on this project by supervising the clinical trials with patients.

References

[1] Patterson D., Etzioni O., Fox D., and Kautz H., Intelligent ubiquitous computing to support alzheimers patients: Enabling the cognitively disabled, *In Proc. of the 1st Int. Workshop on Ubiquitous Computing for Cognitive Aids*, Gothenberg, Sweden, 2002.

[2] Bouchard B., Bouzouane A., and G. S., A Keyhole Plan Recognition Model for Alzheimer's Patients: First Results, *Journal of Applied Artificial Intelligence (AAI)*, vol. Vol. 22, July 2007, pp. 623-658.

[3] Bouchard K., Bouchard B., and Bouzouane A., A new qualitative spatial recognition model based on Egenhofer topological approach using C4.5 algorithm: experiment and results, *Int. Conf. on Ambient Systems, Networks and Technologies (ANT)*, Niagara Falls, Canada, 2011, Elsevier publisher;2011, pp. 497–504.

[4] Capezio F., Giuni A., Mastrogiovanni F., Sgorbissa A., Vernazza P., Vernazza T., and Zaccaria R., Sweet Home! Perspectives of Ambient Intelligence, *In Journal of the Italian AEIT Association*, 2007, pp. 42 - 49.

[5] Geib C. and Goldman R., Partial Observability and Probabilistic Plan/Goal Recognition, *Int. Joint Conference on Artificial Intelligence*, 2005, pp 418-425.

[6] Hartigan J. A., Clustering Algorithms, *John Wiley and Sons Inc.* New York, 1975.

[7] Selim S. Z. and Ismail M. A., K-means type algorithms: a generalized convergence theorem and characterization of local optimality, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, 1984, pp. 81–87.

[8] Cui X., Gao J., and Potok T., A Flocking based algorithm for document clustering analysis, *Journal of Systems Architecture*, Vol. 52, 2006, pp. 505-515.

[9] Bellaachi A. and Bari A., OFLOSCAN : A Flocking-Based Data Mining Algorithm for Detecting Outliers in Cancer Microarrays, 2010.

[10] Reynolds C., Flocks, herds, and schools: a distributed behavioral model, *Computer Graphics*, vol. Vol. 21, 1987, pp. 25-34.

[11] Liu S., Liu Y., Ni L. M., Fan J., and Li M., Towards mobility-based clustering, *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, DC, USA, 2010, pp. 919-928.

[12] Jakkula V. R. and Cook D. J., Enhancing Smart Home Algorithms Using Temporal Relations, *in Technology and Aging.* vol. 21, A. Mihailidis, J. Boger, H. Kautz, and L. Normie, Eds., ed Amsterdam: IOS Press, 2008, pp. 3-10.

[13] Gu T., Chen S., Tao X., and Lu J., An unsupervised approach to activity recognition and segmentation based on object-use fingerprints, *Data Knowledge Engineering*, vol. 69, 2010, pp. 533-544.

[14] Palmes P., Pung H. K., Gu T., Xue W., and Chen S., Object relevance weight pattern mining for activity recognition and segmentation, *Pervasive Mobile Computing*, vol. 6, 2010, pp. 43-57.

[15] Berkhin P., Survey of clustering data mining techniques, *Accrue Software Research Paper*, 2002.

[16] Anderberg M. R., Cluster Analysis for Applications, *Academic Press Inc.* New York, 1973.

[17] Handl J., Knowles J., and Dorigo M., Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1D-SOM, *Technical Report TR/IRIDIA/2003–24*, Universite Libre de Bruxelles, Belgium, 2003.

[18] Deneubourg J., Goss S., Franks N., Sendova-Franks A., Detrain C., and Chretien L., The dynamics of collective sorting: Robot-like ants and ant-like robots, J.-A. Meyer, S. Wilson (Eds.), *Proceedings of the First Int. Conf. on Simulation of Adaptive Behaviour: From Animals to Animals 1*, MIT Press, Cambridge, MA, 1991, pp. 356–365.