# Towards Adaptive Clustering in Self-monitoring Multi-Agent Networks

Piraveenan Mahendra rajah[1], Mikhail Prokopenko[2], Peter Wang[2], Don Price[3]

[1]University of Adelaide, Adelaide, SA 5000, Australia
[2]CSIRO Information and Communication Technology Centre
[3]CSIRO Industrial Physics
Locked bag 17, North Ryde 1670, Australia
contact author: mikhail.prokopenko@csiro.au

**Abstract.** A Decentralised Adaptive Clustering (DAC) algorithm for self-monitoring impact sensing networks is presented within the context of CSIRO-NASA Ageless Aerospace Vehicle project. DAC algorithm is contrasted with a Fixed-order Centralised Adaptive Clustering (FCAC) algorithm, developed to evaluate the comparative performance. A number of simulation experiments is described, with a focus on the scalability and convergence rate of the clustering algorithm. Results show that DAC algorithm scales well with increasing network and data sizes and is robust to dynamics of the sensor-data flux.

## 1 Introduction

Structural health management (SHM) is expected to play a critical role in the development and exploitation of future aerospace systems, operating in harsh working environments and responding to various forms of damage and possible manufacturing and/or assembly process variations. Ultimately, large numbers of sensors will be required to detect and evaluate a wide range of possible damage types within a large and complex structure. Robustness, scalability, reliability and performance verification are also key SHM requirements. A future vision of self-monitoring robust aerospace vehicles includes both local and global SHM systems. The local actions are anticipated to identify, evaluate, and trigger repair for a wide range of damage or defect conditions in aerospace materials and structures. In parallel, global actions should enable dynamic evaluation of structural integrity across large and remote areas. This dual architecture, in turn, entails the need for dynamic and decentralised algorithms used in damage detection, evaluation, diagnostics, prognosis and repair. In order to address these requirements we have chosen to investigate the application of a complex multi-agent system approach to the architecture, and seek to develop methodologies that will enable the desired responses of the system (the remedial actions) to emerge as self-organised behaviors of the communicating system of sensing and acting agents.

CSIRO-NASA Ageless Aerospace Vehicle (AAV) project developed and examined several essential concepts for self-organising sensing and communication networks [1, 8, 3, 9]. Some of these concepts are being developed, implemented and tested in the AAV Concept Demonstrator (AAV-CD): a hardware multi-cellular sensing and communication network whose aim is to detect and react to impacts by projectiles that, for a vehicle in space, might be micrometeoroids or space debris. The CD consists of "cells" that not only form a physical shell, but also have sensors, logic, and communications. Currently, each cell contains a small number of passive piezoelectric polymer sensors bonded to an aluminium skin panel in order to detect the elastic waves generated in the structure by impacts. Each AAV cell contains two digital signal processors, one of which acquires data from the sensors, while the other runs

the agent software and controls the communications with its neighboring cells. Importantly, a cell communicates only with four immediate neighbors. The CD does not employ centralised controllers or communication routers. A stand-alone Asynchronous Simulator capable of simulating the CD dealing with some environmental effects such as particle impacts of various energies has been developed and used in the reported experiments.

Single cells may detect impacts and triangulate their locations, while collections of cells may solve more complex tasks, for example, produce an impact boundary with desired characteristics or an impact network [3, 9] to pre-optimise secondary inspections and repairs. Some responses could be purely local, while some may require emergence of dynamic reconfigurable structures, with some cells taking the roles of "local hierarchs". In fact, most hierarchical clustering architectures for multi-agent networks are based on the concept of a cluster-head (a local hierarch). A cluster-head acts as a local coordinator of transmissions within the cluster. Often, a cluster-head is dynamically selected among the set of nodes. Moreover, clusters would form and re-form when new damage is detected on the basis of the local sensor signals. In the SHM context, an example of a coordinated task initiated by a cluster head is Active Damage Interrogation (ADI) with a piezoelectric transducer array, emitting moderate to high frequency energy from one or more transducers and using the other transducers as sensors to monitor the energy propagation through the structure. A meaningful ADI scenario may require an emergent formation of clusters of cells with similar damage levels. A cluster-head would then fit the data to a diagnostic model. The sensor-data clustering task has two primary challenges:

- Decentralised clustering: most existing algorithms for clustering focus on how to form clusters, given a file or database containing the items. 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 [6];
- Dynamic (on-line) clustering: new events may require reconfiguration of clusters — thus, the resulting patterns or clusters have to be constantly refined.

This requires efficient algorithms for decentralised sensor-data clustering in a distributed multi-agent system. In Section 2 we describe an adaptive algorithm enabling self-organisation of stable but reconfigurable impact data clusters, connecting the cells which detected impacts with energies within a certain band (e.g., non-critical impacts). These clusters are expected to reconfigure in real-time if required. Importantly, the cluster algorithm should be robust in the face of changes caused by new damage, cells' failures and node insertion/ removal. Section 3 presents comparative analysis between decentralised and centralised versions of the developed algorithm, followed by a discussion of the obtained results and future work.

## 2 Adaptive Clustering Algorithms

In this section we describe the decentralised and centralised versions of the developed adaptive clustering algorithm. The input can be described as a series (a flux) of impact energies detected at different times and locations, while the output is a set of non-overlapping clusters, each with a dedicated cluster-head (an AAV cell) and a cluster map of its followers (AAV cells which detected the impacts) in terms of their sensor-data and relative coordinates.

Before presenting details of the algorithms, we would like to position our work in relation to the method of clustering within a fully decentralised multi-agent system, proposed recently by Ogston et al. [6]. The work of Ogston and her colleagues clearly points out the problems of centralised clustering when "data is widely distributed, data sets are volatile, or data items cannot be compactly represented". They present a method for grouping networked agents with

similar objectives or data without collecting them into a centralised database, which shows very good scalability and speed in comparison with the k-means clustering algorithm. The method employs a heuristic for breaking large clusters when required, and a sophisticated technique dynamically matching agents objectives, represented as connections in the multi-agent network. The reported clustering results provide a considerable motivation for our effort. Nevertheless, we need to account for specifics of our application: a particular communication infrastructure where each cell is connected only to immediate neighbours in Von Neumann neighbourhood; constraints on the communication bandwidth; dynamic impact scenarios where density of impacts may vary in time and space; a decentralised architecture without absolute coordinates or id's of individual cells on a multi-cellular aerospace vehicle skin; etc. Therefore, our main goal is not a new clustering method *per se*, but rather an evaluation of a simple clustering technique in a dynamic and decentralised setting, exemplified by the CD sensor and communication network, in terms of scalability and convergence, under specific communication constraints. To this effect, we attempted to abstract away some sensor-data features. For example, instead of considering time-domain or frequency-domain impact data, detected and/or processed by cell sensors [8], we represent a cell sensory reading with a single aggregated value ("impact-energy"), define "differences" or "distances" between cells in terms of this value, and attempt to cluster cells while minimising these "distances". This approach can be relatively easily extended to cases where "distances" are defined in a multi-dimensional space. In short, our focus is on evaluating inter-agent communications required by a decentralised clustering algorithm, dynamically adapting to changes.

The algorithm involves a number of inter-agent messages notifying agents about their sensory data, and changes in their relationships and actions. For example, an agent may send a recruit message to another agent, delegate the role of cluster-head to another agent, or declare "independence" by initiating a new cluster. Most of these and similar decisions are based on the clustering heuristic described by Ogston et al. [7], and a dynamic offset range. This heuristic determines if a cluster should be split in two, and the location of this split.

### 2.1 Clustering Heuristic

Firstly, all $n$ agents in a cluster are sorted in decreasing order according to their impact-energy value $x$. Then, a series of all possible divisions in the ordered set of agents is generated. That is, the first ordering is a cluster with all agents in it; the second ordering has the agent with the largest value in the first cluster and all other agents in the second cluster; and so forth (the $n$-th division has only the last $n$-th agent in the second cluster). For each of these divisions, the quality of clustering is measured by the total square error:

$$E_j^2 = \sum_{i=1}^{k} \sum_{x \in C_{i,j}} \|x - m_{i,j}\|^2 \ ,$$

where $k$ is a number of considered clusters ($k = 2$ when only one split is considered), $C_{i,j}$ are the clusters resulting from a particular division and $m_{i,j}$ is the mean value of the cluster $C_{i,j}$. We divide $E^2$ values by their maximum to get a series of normalised values. Then we approximate the second derivative of the normalised errors per division:

$$f''(E_j^2) = \frac{(E_{j+1}^2 + E_{j-1}^2 - 2E_j^2)}{h^2} \ ,$$

where $h = \frac{1}{n}$. If the peak of the second derivative is greater than some threshold $T$ for a division $j$, we split the set accordingly; otherwise, the set will remain as one cluster.

### 2.2 Decentralised Adaptive Clustering (DAC)

Each agent is initially a follower to itself, and its followers' list will contain only itself. Each agent is also a cluster-head initially (a singleton cluster). The communication messages (shown in *italic*) can be "flooding" broadcasts or dead-reckoning packets using relative coordinates of their destination on the AAV grid. The algorithm involves the following steps carried out by each cell (agent) which detected an impact with the value $x$ (henceforth, references like "larger" or "smaller" are relative to this value):

1. Keeps broadcasting its *recruit* message initially (*recruit* messages will always contain the followers' list of an agent). This broadcasting is done periodically, with a broadcasting-period $P$, affecting all agents with values within a particular offset of the value $x$ of this agent, i.e., with values between $x - \varepsilon$ and $x + \varepsilon$. The offset $\varepsilon$ is initially set to a proportion $\alpha$ of its agent value: $\varepsilon = \alpha x$.
2. If an agent in a singleton cluster receives a *recruit* message from a "smaller" agent, it ignores it.
3. If an agent $p$ in a singleton cluster receives a *recruit* message from a "larger" agent $q$ in a singleton cluster, it becomes its follower, stops broadcasting its own *recruit* messages and sends its information to its new cluster-head $q$: an *acceptance-message* with its relative coordinates and the agent-value $x$. It also stores details of the cluster-head $q$: the agent-value $x_q$ and relative coordinates.
4. If an agent $p$ in a singleton cluster receives a *recruit* message from a "larger" agent $q$ which does have other followers, it ignores the message: simply because the "larger" agent $q$ would also receive and handle a *recruit* message from $p$ itself (see step 6).
5. If an agent receives an *acceptance-message* from some potential follower agent, it adds the agent involved in its followers' list.
6. If a member of a non-singleton cluster, either the head or a follower, receives a *recruit* message (either from a "larger", "smaller" or "equal" agent), it *forwards* it to its present cluster-head.
7. After *forwarding* a recruit message to its cluster-head, a follower ignores further *recruit* messages until the identity of its head has been re-asserted (as a result of the clustering heuristic being invoked somewhere).
8. The cluster-head waits for a certain period $W$, collecting all such *forward* messages (the period $W$, called heuristic-period, should be greater than $2P$). At the end of the heuristic-period, the clustering heuristic is invoked by the cluster-head on the union set of followers and all agents who *forwarded* the messages. The "largest" agent in any resulting cluster is appointed as its cluster-head.
9. The cluster-head which invoked the heuristic notifies new cluster-heads about their appointment, and sends their cluster maps to them: a *cluster-information* message.
10. A cluster-head stops sending its *recruit* messages $P$ cycles before it invokes the clustering heuristic. If it is re-appointed as a cluster-head, it resumes sending *recruit* messages.
11. If an agent receives a *cluster-information* message it becomes a cluster-head. If it was already a cluster-head with a cluster map, it erases that cluster map and accepts the new cluster map. It also *notifies* all its new followers.
12. A follower will periodically get *recruit* messages from its cluster-head. If this does not happen for a while, then it means that this follower is no longer in the followers' list of its cluster-head. Then it will make itself a cluster-head and start sending its own *recruit* messages. The offset of these *recruit* messages will be determined by the offsets it had when it was a cluster-head the last time (not necessarily the same as $\varepsilon$).

Because of the unpredictable timing of the clustering heuristics being invoked in various agents, it is possible that a cluster-head keeps a particular agent as its follower even after its offset $\varepsilon$ has changed and this particular agent is now out of range. To counter this, the cluster-head checks its followers' list periodically and removes agents with values out of range. It is also possible that a cell detects a new impact, possibly increasing the agent-value by a large amount. If this agent was a follower, it immediately becomes a cluster-head and *updates* its former cluster-head. The former cluster-head will delete it from its followers' list.

Depending on the nature of the set of agent values, the offset $\varepsilon$ may be initially too small to reach any other agent. To counter this, an agent periodically (with a period $\Delta$) increases its offsets exponentially until a certain limit: $\varepsilon_{k+1} = max(2\varepsilon_k, \beta x)$, where $\varepsilon_0 = \varepsilon$ initially, and $\beta$ is the limit proportion (e.g., the initial $\varepsilon_0$ may be $0.01x$ and after $5$ periods the offset would become $\varepsilon_5 = 0.32x$). Alternatively, the increase will stop when the offsets of an agent have been reset by the clustering heuristic. When the clustering heuristic is applied, it may produce either one or two clusters as a result. If there are two clusters, the offset of each new cluster-heads is modified. It is adjusted in such a way that the cluster-head of the "smaller" agents can now reach up to, but not including, the "smallest" agent in the cluster of "larger" agents. Similarly, the cluster-head of "larger" agents can now reach down to, but not including, the "largest" agent (the cluster-head) of the cluster of "smaller" agents. These adjusted offsets are sent to the new cluster-heads along with their cluster maps.

### 2.3 Fixed-order Centralised Adaptive Clustering (FCAC)

In order to evaluate DAC, its centralised version was developed. To achieve a congruence between decentralised and centralised versions we define the notion of "reachability" imitating the *recruit* messages. That is, any agent that is within a particular offset of the agent-value of a cluster-head, i.e., with values between $x - \varepsilon$ and $x + \varepsilon$, is said to be *reachable* from that cluster-head. This allows us to essentially replace broadcast messages with simple data-array searches. Direct cell-to-cell messages (e.g., *forwarding*) are replaced with simple data-array operations, such as inclusion, deletion, merge, split, etc. For example, an addition of a follower $p$ is decided when an agent $p$ is "reachable" from a cluster-head $q$, and is accomplished by inclusion of $p$ into an ordered list, headed by $q$.

The Centralised Clustering Algorithm is an iterative process, involving two functions. The first function is a sequential processing of each cluster-head "reaching" to all other agents — analogous to *recruit* messages, while each "reaching" event may trigger an addition of a list of followers to an existing cluster — analogous to *forward* messages. The second function follows the first, and invokes the clusering heuristic by all current cluster-heads, potentially changing the clustering outcome. These functions are repeatedly invoked on the list of agents, and offsets are continuously modified (as described at the end of the previous subsection), until any agent is "reachable" only from a unique cluster-head. Once no cluster-head can "reach" agents that are not its followers, the centralised algorithm terminates.

The quality of clustering is measured by the weighted average cluster diameter [10]. The average pair-wise distance $D$ for a cluster $C$ with points $\{x_1, x_2, \ldots, x_m\}$ is given by

$$D = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} d(x_i, x_j)}{m(m-1)/2} \ ,$$

where $d(x_i, x_j)$ is the Euclidean distance between points $x_i$ and $x_j$. The weighted average cluster diameter for $k$ clusters is given by:

$$\bar{D} = \sum_{i=1}^{k} m_i(m_i - 1)D_i \ / \ \sum_{i=1}^{k} m_i(m_i - 1) \ ,$$

where $m_i$ is the number of elements in the cluster $C_i$ with the pair-wise distance $D_i$. This metric is known to scale well with the size of data points and number of clusters in a particular clustering. It does not, however, account for singleton clusters, while favouring small clusters.

We would like to point out, at this stage, that neither decentralised nor centralised algorithm guarantees a convergence minimising the weighted average cluster diameter $\bar{D}$. In fact, DAC

may give different clusterings for the same set of agent values, depending on the physical locations of the impact points. The reason is a different communication flow affecting the adjustment of the offsets. Each time the clustering heuristic is executed in an agent, its offsets are either left alone or reduced (they are never increased). The scope of agents involved in the clustering heuristic depends on the order of message passing, which in turn depends on the physical locations of impacts. The adjusted offsets determine which agents can be reached by a cluster-head, and this will affect the result of clustering. Therefore, for any set of agent values, there are certain sequences of events which yield better clustering results than others.

The developed centralised algorithm does not simulate all possible sequences of events — hence, the name: Fixed-order Centralised Adaptive Clustering (FCAC). Agent values are entered in a random order, which may not initiate the "best" ordering of events and yield the best clustering for a particular data set. In other words, centralisation of sensor-data is not a guarantee of a superior performance, and processing of all permutations is prohibitive even for a very small number of elements. On occasions, DAC may even outperform FCAC. Nevertheless, randomisation of data proved to be sufficient for the purposes of our comparison.

## 3 Experimental Results

We conducted extensive simulations to compare DAC and FCAC algorithms, with the intention of determining whether DAC algorithm is robust and scales well in terms of the quality of clustering and convergence. The quality of clustering is measured by the weighted average cluster diameter $\bar{D}$. The convergence is measured by the number of times (denoted $H$) the clustering heuristic was invoked before stability is achieved with each data set. This number of times is is chosen as a metric instead of the total time taken in order not to reward DAC for employing essentially parallel computation.

Since we wanted to compare the DAC and FCAC algorithms, we considered the ratio of the weighted average cluster diameters $\bar{D}$ given by FCAC and DAC: $\bar{D}_{FCAC}/\bar{D}_{DAC}$. Similarly, we traced the ratio of numbers of times the clustering heuristic was executed before DAC and FCAC algorithms stabilised with each data set: $H_{FCAC}/H_{DAC}$. Random values were used as agent values. Each resulting data series was approximated with polynomials, and the best fit was selected according to Mallows' criterion [5]:

$$C_p = RSS_p/s^2 - L + 2p\,,$$

where $p$ is the polynomial's order, $L$ is the data sample size, $RSS_p$ is the residual sum of squares, and $s^2$ is given by $RSS_M/(L-M)$, which is the estimate for the residual variance obtained from the complete model with all $M$ regressors included. We used Mallows criterion because predictive and descriptive ability of the model dealing with noisy data is important.

The scalability analysis considered two scenarios. The first scenario kept the AAV grid array size constant, while increasing the number of impacts detected within it. The second scenario, on the contrary, fixed the number of impacts, while increasing the grid size. In other words, the density of impacts was increasing in the first case, and decreasing in the second.

**Effect of Increasing Density.** Figure 1[top-left] shows the weighted average cluster diameter ratio $\bar{D}_{FCAC}/\bar{D}_{DAC}$ against the increasing number of impacts. It illustrates that, while the relative performance of DAC decreases with the number of impacts, it scales well and decreases "gracefully". The relative decrease in performance is linear, and DAC gives comparable performance with FCAC for large data sets. Figure 1[top-right] shows that, as the number of impacts increases, DAC needs relatively more and more executions of clustering heuristics to stabilise than the FCAC algorithm. This is expected, as the clustering heuristic needs to

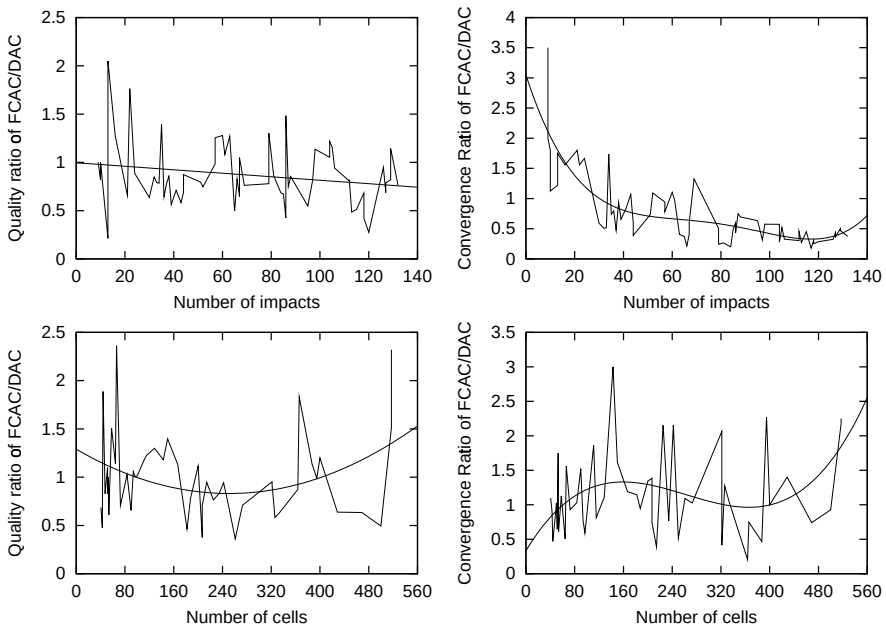


**Fig. 1.** Top-left: the quality ratio $\bar{D}_{FCAC}/\bar{D}_{DAC}$ for increasing impact density. Top-right: the convergence ratio $H_{FCAC}/H_{DAC}$; increasing density. Bottom-left: the quality ratio $\bar{D}_{FCAC}/\bar{D}_{DAC}$; decreasing density. Bottom-right: the convergence ratio $H_{FCAC}/H_{DAC}$; decreasing density.

be invoked in many different agents with limited information in the DAC algorithm. As the number of impacts increases, DAC takes relatively more computation and time before stabilising. However, the best fit (a $4$-th order polynomial) indicates that DAC may in fact stop losing ground with respect to FCAC for even larger numbers of impacts. The reason is that larger data samples have more possible orderings and FCAC has a lesser chance to process the best one. At this stage it is unclear whether DAC will begin to outperform FCAC, but at least it does not perform worse than $4$ times compared to the latter.

**Effect of Decreasing Density.** Figure 1[bottom-left] shows that an increase in the grid size has an interesting effect on the relative performance of the DAC algorithm. Not only the DAC algorithm scales very well with respect to the network array sizes, but it begins to outperform the FCAC algorithm when the array becomes larger. This is indicated by both the best fit (a $2$-nd order polynomial) and the next best (linear) fit. Figure 1[bottom-right] shows that the relative convergence rate of DAC algorithm can be best approximated by a $3$-rd order polynomial (again, the next best fit was linear). Both approximations indicate that, as the array becomes larger, the decentralised algorithm begin to outperform the centralised version.

The described experiments simulated impacts detected at the same time — to evaluate scalability with respect to array sizes and the number of impacts. Another factor is dynamics of the impact flux. To analyse robustness of the DAC algorithm in the face of a spatiotemporal impact flux, we developed scenarios where impacts appear periodically, with varying periods. For the FCAC algorithm, still all data were given at once since, in this case, periodic data insertion is not relevant. Again the ratio $\bar{D}_{FCAC}/\bar{D}_{DAC}$ of weighted average cluster diameters was taken. It was observed that DAC is robust against the timing of impacts: the impact period does not affect the relative performance of the DAC algorithm. In other words, DAC is as good with dynamic data insertion as FCAC is with static (or dynamic) data insertion.

## 4 Conclusions and Future Work

We presented Decentralised Adaptive Clustering (DAC) and Fixed-order Centralised Adaptive Clustering (FCAC) algorithms for self-monitoring impact sensing networks. The experiments indicate that DAC algorithm can be used to cluster sensor-data, achieving a high quality in a dynamic impact sensing network. The DAC algorithm scales reasonably well with respect to array sizes and the number of impacts, and is robust in the face of a spatiotemporal impact flux. This provides a very good support for deploying other, more sophisticated algorithms in the sensing networks. The density-based algorithms may particularly be relevant in our application: e.g., DBSCAN algorithm would allow us to discover clusters with arbitrary shape [2]. Another avenue is, of course, deployment of hierarchical clustering algorithms. However, in this case, rather than parallelising clustering by partitioning the data set over the AAV network and maintaining a central point as done in P-CLUSTER algorithm [4], we are investigating dynamic hierarchies emerging in response to the constraint on the number of clusters, and the communication protocols required for that. In addition, such a dynamic hierarchy in a self-monitoring impact sensing network may be capable of diagnostic actions and localised SHM responses to global patterns detected by the network.

## References


1. Abbott, D., B. Doyle, J. Dunlop, T. Farmer, M. Hedley, J. Herrmann, G. James, M. Johnson, B. Joshi, G. Poulton, D. Price, M. Prokopenko, T. Reda, D. Rees, A. Scott, P. Valencia, D. Ward, and J. Winter. Development and Evaluation of Sensor Concepts for Ageless Aerospace Vehicles. Development of Concepts for an Intelligent Sensing System. NASA technical report NASA/CR-2002-211773, Langley Research Center, Hampton, Virginia, 2002.
2. Ester, M., Kriegel, H., Sander, J., and Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Second International Conference on Knowledge Discovery and Data Mining, 226-231, 1996.
3. Foreman, M., M. Prokopenko, P. Wang. Phase Transitions in Self-organising Sensor Networks. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J.T. Ziegler, J. (Eds.) *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life*, 781–791, LNAI 2801, Springer, 2003.
4. Judd, D., McKinley, P., and Jain A. Large-Scale Parallel Data Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol 20, 8, 871–876, 1998.
5. C. L. Mallows. Some comments on $C_p$, *Technometrics*, vol. 15, 661-675, 1973.
6. Ogston E., Overeinder, B., Van Steen, M., and Brazier, F. A Method for Decentralized Clustering in Large Multi-Agent Systems. Proceedings of the Second International Joint Conference on Autonomous Agent and Multi Agent Systems, 798–796, 2003.
7. Ogston E., Overeinder, B., Van Steen, M., and Brazier, F. Group Formation Among Peer-to-Peer Agents: Learning Group Characteristics. Second International Workshop on Agents and Peer-to-Peer Computing, Lecture Notes in Computer Science series vol. no. 2872, 59–70, Springer, 2004.
8. Price, D., Scott, A., Edwards, G., Batten, A., Farmer, A., Hedley, M., Johnson, M., Lewis, C., Poulton, G., Prokopenko, M., Valencia, P., Wang, P. An Integrated Health Monitoring System for an Ageless Aerospace Vehicle. Proceedings of 4th International Workshop on Structural Health Monitoring, Stanford, September 2003.
9. Wang, P., P. Valencia, M. Prokopenko, D. Price, and G. Poulton. Self-reconfigurable sensor networks in ageless aerospace vehicles. Proceedings of the Eleventh International Conference on Advanced Robotics, 1098–1103, Coimbra, 2003.
10. Zhang, T., Ramakrishnan, R., Livny, M. BIRCH:A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery*, 1(2), 141-182, 1997.