Autonomic Supervision of Stigmergic Self-Organisation for Distributed Information Retrieval

Kieran Greer[1],
Matthias Baumgarten[1],
Maurice Mulvenna[1],
Kevin Curran[2],
Chris Nugent[1],
1. School of Computing and Mathematics,
2. School of Computing and Intelligent Systems,
Faculty of Computing and Engineering,
University of Ulster,
Northern Ireland, UK.

krc.greer;m.baumgarten;md.mulvenna;kj.curran;cd.nugent@ulster.ac.uk

ABSTRACT
This paper will consider how a network of information sources might be autonomously monitored to allow it to self-optimise with respect to querying. While future networks will need to be able to self-adapt, the dynamic and autonomous nature of such networks will make the supervision process more difficult to implement in programming terms. Stigmergic linking is a lightweight and flexible way to provide some form of optimisation. If evaluation functions can also measure the success of any query, then it may be possible to monitor the performance of the self-optimisation. A supervision system could adjust the link update method until an acceptable balance between search time and quality of service is reached. Thus at least in this respect, autonomic supervision would be possible. The monitoring system might also monitor 'concept drift' and detect when it occurs. This measures typical boundaries for concepts of interest and detects when these boundaries are violated. When concept drift occurs, the system would be able to tell if this resulted from a fault or simply a change in the system use and thus be able to apply the appropriate solution.

Keywords
Stigmergy, self-organisation, supervision, self-adaptation.

1 INTRODUCTION
This paper will suggest how a dynamically organised network can autonomously supervise itself to allow it to self-optimise with respect to querying. The philosophy adopted in this paper is to provide a lightweight network structure that can self-organise and adapt using stigmergic and autonomous techniques. The querying mechanism will extract information from a network of knowledge. A 'knowledge network' is a generic structure that organises distributed knowledge of any format into a system that will allow it to be retrieved efficiently (Mulvenna et al. [7], or Baumgarten et al. [1]). The rationale of the knowledge network is to act as a middle layer that connects to a multitude of sources, organises them based on various concepts and finally provides well-structured, pre-organised knowledge to individual services and applications. The knowledge network has been implemented as part of a recent project[1]. To extract the information from the network we need a querying mechanism. Previous work has shown that it is possible to use the query results as part of an optimisation process (Greer et al. [3]). The results of previous queries can be fed back through the network, which will allow nodes that answered similar queries to self-optimise by linking to each other. This will create temporary views that reflect the use of the system. This form of organisation is both autonomous (Menasce and Kephart [6]) and stigmergic (Grassé [2]). By stigmergic it is meant that the network adapts by reacting to its environment, rather than through any particular knowledge and by autonomous it is meant that the network initiates and controls this reaction itself, without the need for external supervision. Introducing autonomous behaviour means that a network can self-configure in environments where supervision is not so easy and can maintain itself, making the system more robust and manageable. Ultimately, query optimisation should reduce the time required to answer a query, but tests have measured the reduction in node count and quality of answer. As the network is service-based, quality of answer has also been called quality of service (QoS). Tests have shown that the linking structure can control the amount of memory that it

BIONETICS 2007, December 10-13, 2007, Budapest, Hungary.

[1] For more information on the overall project see http://www.cascadas-project.org.

uses, thus maintaining a lightweight architecture. There are several parameters that need to be learnt, or specifically set to optimise for a particular configuration. However, memory allocation can, to a certain degree, be monitored by the system itself. Tests have shown that too little or too much memory use can reduce optimisation.

Another aspect of the autonomic network is a supervision system that recognises when faults occur in the system's operation. As part of this, 'concept drift' can be monitored. Concept drift essentially recognises when the values that a part of the system typically returns, drift or change over time. It will be proposed that the supervision system could also monitor the querying performance. This would help the supervision system to determine if the concept drift indicates a fault or a change in the system use. If a change in use, then the supervision system would help the network to self-optimise with respect to querying, by adapting the link update method. If the query performance remained unacceptable, then a fault may have been detected and could be dealt with.

The rest of the paper is organised as follows: Section 2 will outline the querying problem and the need for a querying mechanism. Section 3 will summarise the linking method that has been tested for self-optimisation. Section 4 will discuss the supervision system and how it might supervise the linking process. Finally, section 5 will mention some conclusions on the work.

2 THE QUERYING PROBLEM

The need for an advanced, distributed querying mechanism becomes clearer when analysing the four layers of the knowledge provisioning pyramid as depicted in Figure 1 (taken from Zambonelli et al. [8]).

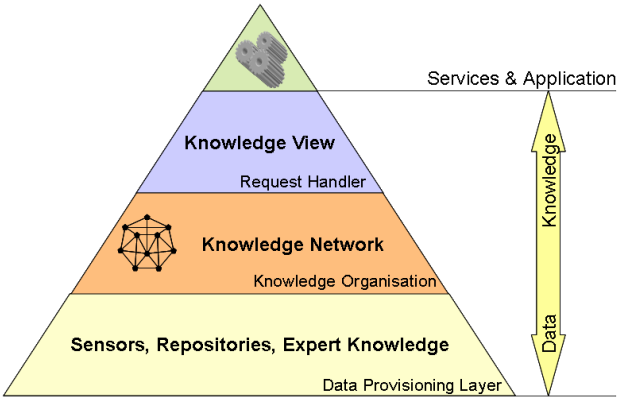


Figure 1. Knowledge Provisioning Pyramid

As seen, raw and unstructured data, which are at the bottom of the pyramid, form the input data for the network. Such data may be pre-processed and pre-organised within the knowledge organisation layer, which forms the core of each network. In order to provide request based and more importantly, highly relevant and well structured knowledge to individual services and applications, which are at the top of the pyramid, a dedicated request layer is required.

Simplified, this request layer is a querying mechanism that searches within the scope of the network for relevant information based on a request object which describes the type and scope of information desired. This query mechanism should take maximum advantage of the relations already built by the knowledge network. However it should not alter them for each request. Therefore, the scope of the querying system is to build a temporary view of individual components without altering the structure of the underlying network. Thus, temporary links that reflect the use of the system can construct these views and can also dynamically change over time. Knowledge discovery concentrates heavily on semantic mapping of ontologies – determining if words are similar based on their meaning or relating them based on prior knowledge. This can be represented by the permanent hierarchical network organisation. There are also relations between elements that are semantically or syntactically very different. This work is innovative in that it does not use the semantic meaning to create the extra relations but the experience of the query search, allowing for semantically unrelated sources also to be linked together. These links are temporary and can complement the permanent organisation provided by the hierarchical network structure and help to create the views of the request layer. Although the realised information will become part of the ontology, it does not really use the ontology to create it and so assumes no prior knowledge.

3 STIGMERGIC LINKING

The network can be considered to consist of a number of distributed nodes. Some nodes link to sources, while some aggregate other nodes, to form a hierarchical structure. The hierarchical structure could be used to largely guide the search, but then dynamic links can further optimise the search process. The links that are generated will be created through stigmergic mechanisms, which in this case means an experience-based method of updating weight values until they reach a certain threshold. The network can automatically send the results of the queries back through itself, where the appropriate nodes will be informed and can update weight values relating to links to other nodes. This paper focuses primarily on source linking, though local views can also be created. Each source can store a structure that records other sources related to it through the querying. This structure can monitor related sources at different levels. Consistent associations between two sources will move the related link up the levels. If the association disappears, the link will move down the levels again, until it is removed completely. The queries that have been tested are of the 'Select-From-Where' type, where the linking is constructed from the sources that satisfy the 'Where' clause evaluations and also link to the sources that answer the query. An example query might be:

Select A.value1, B.value2 From A, B, C where A.value3 LT B.value2 AND B.value4 GT C.value1

The key to the linking structure can simply be a set of keywords that define the part of the query that the link relates to. For example, if the query process evaluates the comparison 'A.value3 LT B.value2', then the B-type source instance can store a link to the related A-type source instance that satisfies the evaluation through a set of key paths defined as:

value2 - A source type – value3 – LT - A source instance.

This thus records only part of a query answer. This query part can be used in different whole queries and is more lightweight and flexible than a caching mechanism. In statistical database systems for example, the system might store the answers to commonly asked queries, so that the answer can be immediately returned when the query is executed. This is particularly useful for queries that require a large amount of processing. The linking mechanism however can re-use parts of other query answers in new queries, making this information retrieval much more flexible. Also, with dynamic or volatile data, the linking mechanism can retrieve current values, when a stored answer may become out of date.

The references in one source to other sources can be stored at different levels in the linking structure, when only the top level references are returned as links. It is these top level references that will be looked at for answers when the appropriate query is executed. The top level references will be returned as possible sources to look at rather than requiring the network to look at all potential sources. The optimisation principle is very simply illustrated in Figure 2. Consider a network where there are sources of the type A or B. Each type has a number of instances. Maybe there are two types of sensor, with many sensors in total providing different readings. Two sources instances – A1 and B1 – are typically used to answer a particular query comparison. Thus a link builds up between them. When the same query part is evaluated again, the linked search can retrieve just the A1 source when the B1 source is used. A full search however would need to look at all of the A-type sources, for example A2 as well. Thus some optimisation has been achieved.

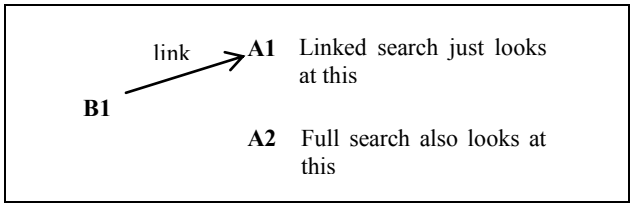


Figure 2. Example of a source link.

The allowed amount of memory, or number of references to linked sources, can be controlled, thus ensuring that the structure stays lightweight. Source references are stored with weight values that can be incremented or decremented depending on their subsequent use. This will move the references up or down a level when they fall above or below the related level threshold. The association between the two linked sources thus needs to be consistent, that is, when one is used in a query part the other is also always used. If this is not the case then the link weight value will decrease. However, controlling memory means that there is some juggling of the movement of source references. If a particular level is full, then it will need to remove a reference before another can be added, and so on

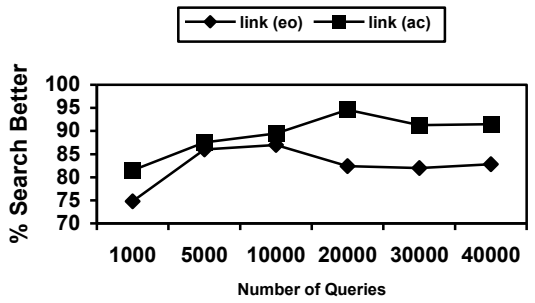
The linking mechanism for these tests also included a local view that could also be retrieved to help to optimise the search process. This could represent a view created by a particular application to reference the nodes that it typically used. If a particular application typically only used part of a whole network, then it could create a local representation of the nodes that it typically visited. This would be specific to the application, as opposed to the source links of the global network that any user's query would create. The local view would further optimise the retrieval process by further reducing the search space to certain sources.

Queries that used the equivalence comparison only (eo) or all comparison operators (ac) were tested. While these tests used numerical values, the equivalence only queries would also be useful for text or concept matching. Comparing two concepts can apply to numbers or text equally. The queries were tested on a random network with 10 source types and 30 instances of each type. Each instance had 5 value types, each with a random value in the range 1 to 10. For the linking to be effective it is assumed that consistent types of query need to be executed. The queries were skewed so that 90% of the time one of 3 source types or 2 value types would be selected and 10% of the time one of the remaining 7 source types or 3 value types would be selected.

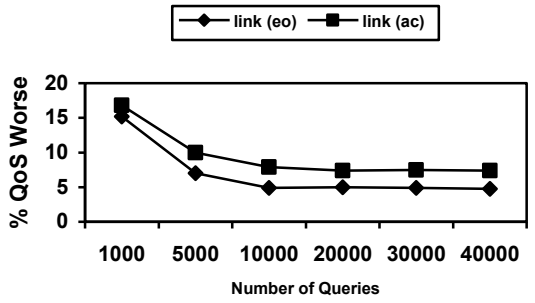
Graphs 1 and 2 show the potential reduction in node count and the related depreciation of QoS, when links are used to answer the queries. Note that the querying mechanism included the use of a view. An evaluation function tried to maximise the total answer value (sum total for all sources in the answer). This was simply used as the factor to distinguish what the best answer was. It would thus select certain nodes that the linking mechanism would then need to recognise and link. If the correct nodes were linked then the linking mechanism would be a success. This could be measured by comparing the sum total for a full search with the sum total for a linked search. If the linking mechanism did not link the correct nodes, then the linked search would return a smaller total and this deprecation is illustrated in Graph 2.

Tests showed that in general, a larger search would produce a better QoS. This would be expected as the larger search can look at more potential sources. However, if the links added were key, then both factors could be improved. The results showed that the equivalence only queries performed the best with regard to QoS, but Graph 1 shows that with regard to search reduction, it is also possible to have too many links. Performance can be measured as a balance between search reduction and QoS. If however the search increases without a corresponding improvement in QoS, then there may be too many links. Graphs 1 and 2 suggest that there are maybe too many links from 2000 queries (eo) or 30000 queries (ac) onwards. Graph 3 shows the average number of links per

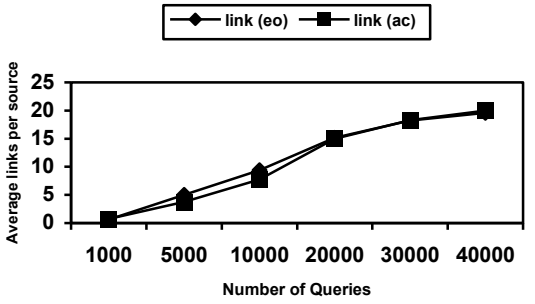
source, where the maximum number of allowed links was 50. Note that this does not reflect the true distribution, when some sources would have more links than others, but an average value. These results suggest that increasing numbers of queries might produce too many links or contradicting links. Thus there may be a need for monitoring, to determine when the upper limit on performance has been reached.



Graph 1. Percentage of reduction in the number of nodes searched for queries with the equivalence operator only (eo) or all comparison operators (ac).



Graph 2. Percentage of reduction in quality of service for queries with the equivalence operator only (eo) or all comparison operators (ac).



Graph 3. Average number of source links stored for queries with the equivalence operator only (eo) or all comparison operators (ac).

4 THE SUPERVISION SYSTEM

There is strong motivation for new perspectives on generic supervision methodologies in order to provide more resilience in the face of ever more complex systems. In particular, future autonomic systems will need to supervise the dynamic aggregation of individual autonomous working components. The complexity and size of future systems will make manual supervision impracticable. While the effective real-time coordination and management of such systems forms another obstacle, the pervasive supervision of such services as well as the underlying environments, can be seen as the final step towards truly and continuous ubiquitous and pervasive computing. Furthermore, the dynamic nature of such systems may mean that usage or values will change over time. This will continuously open a gap between the actual model and the real world concept they were designed for. This problem, referred to as 'concept drift', implies the constant adaptation of intelligent services and their underlying models in order to achieve a stable state around some pre-defined boundaries and to prevent a system becoming unstable by drifting outside of its operational parameters[2]. The idea of concept drift is that the characteristics of certain attributes change with time. Stemming from the area of predictive analytics, a concept of interest can be defined by its underlying contextual information or raw data.

Liao et al. [5] also consider the problem of fault detection through recognising anomalies. Anomaly detection analyses a set of characteristics of the monitored system (or users) and identifies activities that deviate from the normal behavior. It is assumed that such deviations may indicate that an intrusion or attack exploiting vulnerabilities has occurred (or may still be occurring). Any observable behavior of the system can be used to build a model of the normal operation of the system. System and network changes however could also occur from legitimate reasons, for example, simply a change in the system use. Therefore, the normal behavior may not be strictly predictable in the long term. This problem is known as concept drift in machine learning literature. An effective supervision system would detect concept drift, but also be able to determine the cause (fault or change in use) and adapt as appropriate. If the correct action is not taken, a large amount of false alarms would be generated if the normal behavior model failed to change accordingly to accommodate the new values.

As Kubat and Widmer [4] describe, in the problem of on-line learning, the essence is to make the learner recognise gradual or abrupt changes in the target concept and adjust accordingly the internal representation of the concept. Such changes are usually referred to as concept drift and can be caused by a changed context. One system capable of tracking concept drift is FLORA (FLOating Rough Approximation). This considers only a relatively recent set of values, called the 'window', and derives three groups of symbolic descriptions from them. These are for windows with all positive examples, some positive examples or negative examples with no positive examples. Heuristics

[2] On-going work by Baumgarten.

have since been added to this system to allow it to adapt to abrupt drifts. The system described in Kubat and Widmer [4] is called FRANN, which uses a Radial Basis Neural Network to measure the windows. While most drift systems have symbolic data in mind, most realistic concepts can only be described by numerical or mixed numerical/symbolic representations. Symbolic learners can thus split the numerical ranges into intervals and represent each interval with a boolean variable.

A potential area where concept drift may occur with regard to querying is when the use of the system changes. For example, if the system is typically queried about certain concepts, then it might typically provide certain answers that also associate certain nodes with each other. If over time the queries on the system changes, then it will provide different answer sets that might require a different set of links. This would mean that the currently monitored values and links would be inappropriate and should be updated. A supervision system could recognise this by the fact that the existing links are returning a QoS or overall performance that is worse than normal. If we consider the source values to be too erratic for monitoring, then we could consider aggregated values at other nodes in the system. For example, a number of sensors might all return their values to a higher level node that then averages them. If these are being monitored as part of concept drift, then the change can be detected at this level also.

4.1 The Proposed Supervision Solution for Monitoring Query-Optimisation

So the supervision system will measure concepts of interest and recognise when they start to fall outside of their boundaries. The precise implementation of the concept drift component is not of interest, but would be based on existing models to monitor symbolic or numerical data. At the same time the linking performance is being monitored with respect to querying. Concept drift is recognised and so the system wonders if it has detected a fault. It checks the querying performance and recognises that there has recently been a change in performance. It thus modifies the link update method to try and improve this. Performance improves to a satisfactory level but the concept drift remains. Thus the system recognises that the system use has changed and it needs to update its drift parameters. It modifies its drift parameters to fit the new data being fed back.

Later on the system recognises concept drift again, coupled with a change in query performance. Query performance does not improve and so the system now needs to check if certain services are now making the wrong calculations and returning incorrect values. It notifies the self-heal component so that it can fix the fault if one exists. Another aspect of this is that, in an open system where services can be loaded and used by anyone, inconsistent behaviour may occur. A service could start to return the wrong results in purpose, in order to make it more attractive to other services. If this service started answering queries with values that were abnormally good, this inconsistency could be detected and appropriate action taken.

Figure 3 suggests a generic algorithm that might be tried as part of a monitoring process. This would monitor concept drift and QoS, detect when a change occurs and take appropriate action. For this algorithm, QoS includes quality of answer and search time.

```
While (system being supervised)
    Evaluate the last query result (of some type using some
    evaluation method)
    If (CD  OR QoS deprecation) Then
        If (QoS deprecation) Then
            If (QoS too bad)
                Notify self-heal component to take action.
            Else
                If (first update or recent updates indicate
                improving QoS) Then
                    Only allow link decrements with no
                    increments. Do not allow new link
                    additions.
                Else If (QoS not improving)
                    Only allow link decrements with no
                    increments of existing links, but also
                    allow new links to be added. Try to
                    replace existing links as quickly as
                    possible.
                End If
            End If
        Else If (CD)
            If (QoS too good)
                Perform check on suspicious component.
                If (component OK)
                    Adjust concept drift parameters to
                    represent the new situation.
                Else
                    Notify self-heal component to take
                    action.
                End If
            Else
                Adjust concept drift parameters to represent
                the new situation.
            End If
        End If
    Else If (QoS too good)
        Perform check on suspicious component.
        If (component NOT OK)
            Notify self-heal component to take action.
        End If
    Else
        Allow link updating as normal (increments and
        decrements)
    End If
End While
```

Figure 3. Generic Link Monitoring Algorithm

Current tests have measured queries with numerical values and so at least in this context, an optimising evaluation

function exists. While the tests have measured the node count and QoS, the supervision system might measure search time with QoS. Thus, as extra links are created then the search time will be increased. If this increase in time does not produce any improvement in QoS, then the system can be considered to have too many links and is not properly optimised. Link changes due to change in use will occur naturally over time, but a speeding up of this process may be required to recover acceptable performance in an acceptable amount of time. While evaluation functions should exist for numerical data, for textual data it may be more difficult. But if an evalution function can be clearly defined, a supervision system can be successfully deployed.

You can argue that measuring every parameter of every query is far too complex to be possible. However, there is no clear rule about any monitoring system and what it can measure. We do not know the exact number of parameters, size of network, or how they will eventually be evaluated. Maybe a service monitors itself, or maybe further up the hierarchy aggregated evaluations take place. This is a general problem for all concept drift monitoring.

5 CONCLUSIONS

This paper has considered how a network of information sources might be autonomously monitored to allow it to self-optimise with respect to querying. Query performance can be measured as a factor of the search time and the quality of answer. A general rule would be that a larger search time would produce a better quality of answer, but for a practical network some balance must be met to allow both parameters to reach acceptable values. An exact equation to measure this balancing is still an open question. As a start, you could maybe notice when one parameter stays the same while another deprecates, indicating a problem. While future networks will need to be able to self-adapt, the dynamic and autonomous nature of such networks will make the supervision process more difficult to implement in programming terms. Stigmergic linking is a lightweight and flexible way to provide some form of optimisation. If there exist evaluation functions to measure the success of any query, these processes can provide a practical way to allow the network to self-optimise and also monitor the performance of that optimisation. Thus at least in this respect, autonomic supervision should be possible. Another aspect of supervision is concept drift. This can be used to detect when a fault occurs in the system that causes it to produce irregular values, outside of current concept boundaries. However, a valid change in system use could also cause this event and so combining the QoS monitoring with the concept drift monitoring will allow the system to determine when concept drift indicates a fault or when there has simply been a change in system use.

While this paper considers stigmergic links created from query or search results, the links could be stigmergically created through any viable mechanism and simply require configuration parameters and an evaluation function to measure their effectiveness. The linking structure proposed for the query-related links is very simple and could be adapted to many different scenarios.

6 ACKNOWLEDGEMENTS

Work supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

7 REFERENCES

[1] Baumgarten, M., Bicocchi, N., Curran, K., Mamei, M., Mulvenna, M.D., Nugent, C., Zambonelli, F. Towards Self-Organizing Knowledge Networks for Smart World Infrastructures, *Invited Session on Service Development and Provisioning through Situated and Autonomic Communications at International Conference on Self-Organization and Autonomous Systems in Computing and Communications (SOAS'2006)*, Erfurt, Germany, (18 - 21 September, 2006).

[2] Grassé P.P. La reconstruction dun id et les coordinations internidividuelles chez Bellicositermes natalensis et Cubitermes sp., La théorie de la stigmergie: essais d'interprétation du comportment des termites constructeurs, *Insectes Sociaux*, 6, (1959), 41-84.

[3] Kieran Greer, Matthias Baumgarten, Maurice Mulvenna, Kevin Curran and Chris Nugent. Knowledge-Based Reasoning through Stigmergic Reasoning, *International Workshop on Self-Organising Systems IWSOS'07, In: David Hutchison and Randy H. Katz (Eds.), Lecture Notes in Computer Science, LNCS 4725, Springer-Verlag*, (11 – 13 September 2007), 240 - 254.

[4] Miroslav Kubat and Gerhard Widmer. Adapting to Drift in Continuous Domains, *Proceedings of the 8th European Conference on Machine Learning, Lecture Notes in Computer Science, LNCS 912*, (1995), 307 – 310.

[5] Yihua Liao, V. Rao Vemuri and Alejandro Pasos. Adaptive anomaly detection with evolving connectionist systems, *Journal of Network and Computer Applications, 30*, (2007), 60-80.

[6] Menasce D.A., Kephart., J.O. Autonomic Computing, *IEEE Internet Computing*, (2007), 18 – 21.

[7] Mulvenna, M.D., Zambonelli, F., Curran, K., Nugent C.D. Knowledge Networks, *In: I. Stavrakakis and M. Smirnov (Eds.), Autonomic Communication, Springer-Verlag, Lecture Notes in Computing Science, LNCS 3835*, (2006), 99-114, ISBN 3-540-32992-7.

[8] Franco Zambonelli, Matthias Baumgarten and Nicola Bicochi, Deliverable D5.1: Knowledge Networks Specifications, Mechanisms, and Alpha Software Release, *IST FET European Framework VI Cascadas Project*, (2007), http://www.cascadas-project.org, (last accessed 10/10/07).