Emergent Service Provisioning and Demand Estimation through Self-Organizing Agent Communities

Mariusz Jacyno
School of Elec. & Comp. Sci.
University of Southampton, UK
mj04r@ecs.soton.ac.uk

Seth Bullock
School of Elec. & Comp. Sci.
University of Southampton, UK
sgb@ecs.soton.ac.uk

Michael Luck
Dept. of Computer Science
Kings College London, UK
michael.luck@kcl.ac.uk

Terry R. Payne
Dept. of Computer Science
University of Liverpool, UK
T.R.Payne@liverpool.ac.uk

ABSTRACT

A major challenge within open markets is the ability to satisfy service demand with an adequate supply of service providers, especially when such demand may be volatile due to changing requirements, or fluctuations in the availability of services. Ideally, this supply and demand should be balanced; however, when consumer demand changes over time, and providers independently choose which services they provide, a coordination problem known as 'herding' can arise bringing instability to the market. This behavior can emerge when consumers share similar preferences for the same providers, and thus compete for the same resources. Likewise, providers which share estimates of fluctuating demand may respond in unison, withdrawing some services to introduce others, and thus oscillate the available supply around some ideal equilibrium. One approach to avoid this unstable behavior is to limit the flow of information between agents, such that they possess an incomplete and subjective view of the local service availability.We propose a model of an adaptive service-offering mechanism, in which providers adapt their choice of services offered to consumers, based on perceived demand. By varying the volume of information shared by agents, we demonstrate that a co-adaptive equilibrium can be achieved, thus avoiding the herding problem. As the knowledge that agents possess is limited, they self-organise into community structures that support locally shared information. We demonstrate that such a model is capable of reducing instability in service demand and thus increase utility (based on successful service provision) by up to 59%, when compared to the use of globally available information.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multagent Systems

General Terms

Algorithms, Design, Experimentation

Keywords

Self-organising, service provisioning, adaptive service markets

Cite as: Emergent Service Provisioning and Demand Estimation through Self-Organizing Agent Communities, Mariusz Jacyno, Seth Bullock, Michael Luck and Terry R. Payne, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

1. INTRODUCTION

Ensuring a good balance in the level of supply of services within a Multi-Agent System is essential in avoiding the failure of tasks due to service unavailability, or a drop in the quality of service due to the over-provision of services from a small number of service providers. Mechanisms that achieve this in large, dynamic and open systems must be scalable, reliable and adaptive to environmental change. It is, however, far from obvious how best to design the individual agents such that the multi-agent system performs efficiently and robustly as a whole. One specific problem facing decentralized systems is the emergence of undesirable behaviors, such as those that lead to service competition [8], which can occur when consumers attempt to access a limited subset of resource providers, leaving others under-utilised. In the absence of centralized control, such behaviours may be self-reinforcing, leading to a rapid degradation of system performance.

The El Farol Bar problem [1] demonstrated that instability can emerge within dynamic environments, when independent, rational agents have access to global information. This can be especially problematic when agents all employ similar decision making mechanisms, such as in coalition formation [11], group problem solving [14] and teamwork [9]. A common property of such models is their reliance on distributed protocols and intelligent algorithms that coordinate agent behaviour. However, these mechanisms, generally assume up-to-date shared information, and thus to converge on an optimal solution, they require a substantial amount of global system information (and corresponding interactions) to maintain awareness of peer goals, actions, etc. As a consequence, they are often vulnerable to increasing system scale and/or dynamism [4].

In contrast, there is an emerging interest in the use of the different self-organising properties observed within natural systems, such as insect colonies [10, 6, 2]. There are many different agent design mechanisms that can affect system performance. Provious studies have explored the role of system heterogeneity brought about by limiting knowledge [10] or using decision procedures that diversify agent behaviour [6]. This paper presents a model of a decentralized multi-agent system in which information about the availability of service providers is shared between agents, and the demand for particular services changes over time. Agents must therefore organise the information that is available to them locally in order to adaptively respond to changes in demand in an efficient manner. This is achieved by varying the amount of information available to agents, which affects the flow of information within the system.

An evaluation of our model demonstrates that when agents pos-

sess only a limited awareness of their peers within their neighborhood, they exhibit self-organising behaviour resulting in the emergence of community structures that support locally shared information. By limiting the quantity of information shared (by limiting the knowledge an agent uses), stable local community behavior emerges that is robust and efficient, and also adaptive (without exhibiting any inefficient, oscillatory behaviour) when the agent population is exposed to global fluctuations in service demand.

The paper is organized as follows: Section 2 provides the motivation for our model, and describes much of the related work relevant to the problem. In Section 3, the proposed model is presented in detail, and the behavior of the system in a dynamic environment is then evaluated and analysed in terms of information flow in Section 4. The strengths and limitations of this approach are then discussed (Section 5), before concluding in Section 6.

2. MOTIVATION AND RELATED WORK

Arthur introduced the El Farol Bar problem as a game-theoretic example of the challenge facing rational agents that wish to organize themselves efficiently on the basis of shared information [1]. In this game, each agent wants to visit the bar only if less than 60% of the agent population also wished to visit the same bar. A rational agent that decides to visit the bar would reasonably assume that every other agent would reach the same decision and also choose to visit the bar, in which case it should reverse its decision. However, it could also reason that every other agent would rationalise the problem in the same way. This quandary rests on two symmetries: (i) every agent employs the same decision-making mechanisms, and (ii) every agent reasons on the basis of the same information. Both these symmetries are typically present in many collaborative, decentralized agent systems [11, 14, 9].

A small number of studies have explored the use of agent organization without the assumption of access to global information [10, 6, 2]. In these systems, agents use decision algorithms that operate on local information. The specific problem of resource allocation (as opposed to service provision[1]) in decentralized multi-agent systems has been discussed by Sen *et al.*, who consider a system of self-interested agents allocating resources on the basis of limited knowledge about the global system state [10]. In this context, Sen *et al.* investigated the effects of limiting the agents' access to both knowledge about the state of system resources, and the resulting outcome on system resource utilisation. Hogg and Huberman [6] examine the effects of local decision making on resource utilisation within a computational ecosystem represented by a population of resource allocating agents. The authors demonstrate how imperfect information about resource state can lead to chaotic system behavior and how this can be suppressed through appropriate local decision-making mechanisms. Brueckner and Parunak present a further strategy relying on local learning mechanisms designed to preserve energy-minimising resource allocation strategies within a mobile ad-hoc network [2]. In finding such localized strategies that reconfigure the allocation of resource in a manner that minimises power consumption, the authors draw their inspiration from self-organising properties of natural systems.

There are still many questions that should be explored to understand how local mechanisms interact to produce global behaviour, with the ultimate aim of designing autonomous agents such that their behavior self-organises to deliver efficient resource allocation in a dynamic environment. One approach for interpreting and analysing the behaviour of decentralized systems is in terms of information flow [8, 13], which can provide a window into the efficiency of a system's organisation. At one extreme, there is little to no flow of information between agents who consequently have to act on the basis of extremely limited information. In such cases, consumers tend to make poor provisioning choices, resulting in a difficulty in accessing desired services. Service providers may also have little idea about which types of service are in demand, and hence have little evidence to support strategies for accepting or committing to service requests. At the other extreme, where information can flow freely, or may be globally available, system behavior may become extremely dynamic and unstable (as in the case of Athur's El Farol Bar problem [1]), thus risking the possibility of cached information becoming stale. This can result in a continual change in strategies used by providers to offer services, or the unexpected withdrawal of some service types to facilitate the availability of others in resource-bound environments. In response, provisioning services by consumers can become increasingly challenging, resulting in delays (or even failures) in completing tasks. Between these two extremes, there may exist a regime where information flow amongst agents brings about stable 'communities' within which consumers exhibit consistent resource demand and providers are able to match this demand reliably. When changes in demand occur, a system must be able to reconfigure gracefully, either by agents 'migrating' between communities, or by providers switching which services they offer in order to balance supply.

3. LOCALIZED SERVICE DEMAND MANAGEMENT

In this section, we propose a framework for localized service demand management, which assumes a multi-agent system comprising of provider agents (*providers*), who offer services of a specific type, and consumer agents (*consumers*) which request and utilize the available services to achieve some task. We assume that both service providers and service consumers are agents running on *constrained* hardware components. Depending on the characteristics of the system, interaction between agents may be limited by power consumption (e.g., sensors), bandwidth consumption, or time-delayed response, all of which may have associated costs if service execution must take place "on-demand", quickly or by some deadline. This is represented in the form of service capacity, such that each agent may only satisfy a service request for a limited number of consumers simultaneously. To facilitate the supply of a variety of different services types, we assume an agent is capable of reconfiguring the service type it provides at run-time. This can involve a significant cost in the form of down time during which the new service type is loaded. We also consider that providers increase their utility by successfully providing services, and that consumers increase their utility by successfully consuming services. Thus, to maximise utility, provider agents try to avoid offering services for which there is little demand, and consumer agents reduce wasted time where their requests are not satisfied, by locating providers that can satisfy their requests.

To explore how stable communities emerge, this framework limits the information flow, and the amount of knowledge maintained by different agents. By removing these limits, global models of service demand can be achieved such that all the agents have complete awareness of the current service demand (i.e. similar to the conditions of the El Farol Bar problem described earlier). This is also equivalent to assuming the presence of a single *Matchmaker* [15] which maintains a registry of all available service providers within a multi-agent system. However, by reducing the information flow,

[1]By *service provision*, we refer to the selection by a consumer of some provider agent that can provide some service for that consumer [12].

stable community behavior should emerge whereby local service demand is satisfied by the local providers, thus avoiding the problems associated with global knowledge in dynamic environments.

The individual agents must therefore evolve local policies that facilitate the emergence of this community behavior. Thus, service management and provisioning strategies should emerge from local co-adaptations of individual agents, based on observations of previous transactions, and other information shared by the local community. Consumers exchange information with known providers about their individual service demand and their knowledge of other service providers. The providers then aggregate this information, and both utilize it to determine future service offerings (see below), and to provide community information back to those consumers that they transact with. Whilst this naturally involves sharing some knowledge, the agents independently modify their individual models of the local environment. Since service availability can fluctuate as a result of several factors, including: current demand; contention for services; and demand for other service types (thus resulting in a reconfiguration of service offerings); it is necessary that agents maintain an accurate model of the environment by maintaining a continuous flow of pertinent information with their peers.

As the number and requirements of consumers cannot be known *a priori*, and as this demand and preference will vary over time, the design goals are three fold:

1. which providers should be configured to offer what service types, in order to satisfy current demand;
2. which of the providers known to a consumer should be utilised such that the system minimises competition; and
3. how should the system be organised such that it is robust to changes in supply and demand for particular service types.

3.1 Model for Service Consumers

Consumers are agents that request and consume services provided by one of the provider agents. An agent may be capable of both offering services to its peers as well as consuming services offered by its peers; however, for simplicity, we consider the model for each behavior as separate. The consumer monitors the behavior of known service providers locally, and uses this knowledge both to provision future service requests, and to share this knowledge when establishing community knowledge. The consumer maintains a local registry, $\mathcal{R}_c$ containing tuples corresponding to services that it is aware of. Each tuple is defined as follows:

$$r_p = \langle \alpha_p, type, bias \rangle$$

where α_p corresponds to an agent that has provided the service $type \in Capability$ at some point in the past, and $bias \in [0..1]$ corresponds to a score or preference for using provider α_p to provide a service of type $type$ in the future. As consumers will not possess complete knowledge about whether a provider is currently available, or even if it is still configured to provide the service of type $type$, it periodically updates the tuples in its local registry $\mathcal{R}_c$. This registry is also used when provisioning services of a given type; the consumer ranks all the tuples maintained by the registry relating to a desired service type in decreasing order of $bias$, and then submits a request to the providers in turn (starting with that which has associated the highest $bias$ value), until a provider is found which can satisfy the request. Each request takes some finite time (T_q), and the provider responds either to confirm that it will satisfy the request or to reject it; either because it currently does not provide that service type, or because it is unavailable (i.e. it is satisfying another query and does not have sufficient resource to simultaneously honor an additional request without compromising current commitments).

Typically, once a service request has been satisfied (or if no requester could be found) then the agent exchanges local knowledge with known providers (described below), before becoming inactive for some randomly determined period. The knowledge exchange is assumed to take some finite time, T_i (irrespective of the number of providers involved), and corresponds to the process of sharing information about local service demand (and availability) thus facilitating the evolution of a localized community structure. The inactive period corresponds to those periods in other scenarios (or frameworks) where agents may be performing other tasks, or interacting with a user, and is drawn from a uniform distribution $[0, \omega]$, after which the allocation process begins again.

Consumer stress (c_s) is a measure of local consumer dissatisfaction, and reflects the difficulty an agent may experience when trying to provision a given service type, based on its registry, $\mathcal{R}_c$. The agent maintains an upper limit, f^{max} of request attempts for each task. The motivation is that once the agent has traversed all relevant entries within its registry, previously unavailable providers may subsequently become available. As the registry may be traversed several times whilst attempting to provision a service, this process is terminated after f^{max} failed requests.

During each provision attempt, the agent generates a failure quotient f_t for provision t, based on the number of failed requests, f_r, where $[0 \leq f_r \leq f^{max}]$, defined as follows:

$$f_t = max\left(1, \frac{2f_r}{f^{max}}\right)$$

The intuition here is that the failure quotient should approach unity as the number of failed queries reaches $\frac{f^{max}}{2}$, despite the fact that it can still continue to issue requests before reaching f^{max}. The stress parameter, $c_s \in [0..1]$, is based on the average failure quotient for the current and previous provision, as follows:

$$c_s = \left(\frac{f_{t-1} + f_t}{2}\right)^2$$

The consumer periodically updates the ordered set $\mathcal{R}_c$ to reflect its experience in provisioning services, and to minimize the number of future rejected queries. If the request was successfully satisfied, then the tuple r_p corresponding to the provider α_p which provided the service $type$ is modified, such that $bias$ is incremented by δ_{bias}; otherwise it is decremented by this amount[2]. To ensure that this model of provider availability does not become stale, a decay function is used to adjust the $bias$ parameter for all tuples in $\mathcal{R}_c$, by applying a *decay coefficient*[3]. After each update, the set $\mathcal{R}_c$ is then ordered with respect to $bias$.

3.2 Model for Service Providers

Providers model the local demand for services to determine what services they should offer. However, within resource-bound environments, providers may only offer a limited number of services at one time, despite posessing the *capability* of offering several types of services; due to limitations in physical resources (e.g. memory size, processor capacity, etc), or based on security issues. Business sectors (such as the E-Business sector) also limit the number of software modules that servers can provide to avoid information leak. The suspension of availability of one service type and introduction of another can have an implicit cost, as this reconfiguration

[2]The value $\delta_{bias} = 0.1$ was found empirically.

[3]The decay coefficient used in this model has the value $\delta_{decay} = 0.9$; this value was determined empirically.

typically takes some time during which the agent cannot perform any further service execution, and thus will not obtain any utility increase. We therefore assume that each provider agent α_p can only offer one service type at any time, but has the capability of offering several other service types (subject to reconfiguration). The set $Capability = \bigcup_{\forall \alpha_p \in MAS} Capability_{\alpha_p}$ contains the union of all service types available from all service providers in the multi-agent system (MAS), whereas $Capability_{\alpha_p}$ corresponds to the set of services that α_p is capable of offering. Thus, to determine which service type α_p should offer, it maintains a model $\mathcal{S}$ of current, local service demand, and determines what services to offer from that model. To achieve this, the provider maintains a number of registries corresponding to provider ratings for different service types, i.e. $\mathcal{S}_{type}$ for each registry of type $type$, as provided by consumers during information exchange. Providers receive requests from consumers in the following form:

$$req_i = \langle \alpha_c, type, capacity, c_s \rangle$$

whereby α_c corresponds to the consumer which submitted the request, $type \in Capability$ corresponds to the type of service the consumer requested, the size (in terms of $capacity$) of the task required, and the consumer's current stress, c_s (defined in Section 3.1). When a new request is received that can be satisfied, the provider's registry is also updated. Each request is augmented with a $bias$ rating, and stored as the typle r_c, which is in a form that can also be used when exchanging information with consumers; i.e.

$$r_c = \langle \alpha_c, type, capacity, c_s, bias \rangle$$

If the same consumer for the same service type as that in the request had appeared in a previous request, then the corresponding tuple in $\mathcal{S}_{type}$ for the provider itself is updated with the new $capacity$, and the $bias$ is modified, based on the product of the consumers stress, c_s and δ_{bias}, as follows:

$$bias = max(1, bias + \delta_{bias} c_s)$$

This adjustment reflects an increase in perceived demand for the service type. If no previous request exists from the requester for this service, then a new tuple is added to $\mathcal{S}_{type}$, with an initial $bias = \delta_{bias} c_s$. The union of all the sets, $\mathcal{S}$, is then ordered[4] with respect to $bias$.

The provider periodically consults the ordered set $\mathcal{S}$ to determine whether or not to reconfigure its offered service. If the $type$ of service in the first tuple corresponds to the service that is currently being offered, then no action is taken. Otherwise, the provider performs a *switch* operation, whereby the provider changes the type of service it can provide[5]. To ensure that the model maintained for current service demand does not become stale, a decay function is used to adjust the $bias$ parameter for all tuples in the sets $\mathcal{S}_{type}$ for each type, using the decay coefficient δ_{decay}.

3.3 Facilitating Information Exchange

To facilitate the migration of knowledge about the availability of services, and the current service demand, both providers and consumers share knowledge, before revising their respective models.

3.3.1 Sharing knowledge with Providers

Each consumer shares all of the tuples contained in its local registry, $\mathcal{R}_c$ and its current stress level, c_s, with each of the providers that are listed in the registry. Each provider then uses an integration policy to incorporate this knowledge into its own local registries ($\mathcal{S}_{type}$ for each of the service types the provider knows about). This integration policy limits the number of tuples merged from the consumer's knowledge for each given type (i.e. tuples from $\mathcal{S}_{type}$) with its own knowledge, based on c_s. This stress level indirectly represents the quality of knowledge the consumer possesses; low c_s values suggest that the tuples provide an accurate representation of the current availability of services, whereas high c_s values suggest that the knowledge is poor, thus leading to difficulties in provisioning services. The maximum number of information tuples the provider is willing to substitute ($t_n \in \mathbb{Z}$) for each service type $type$ is defined as follows:

$$t_n = (1 - c_s) * m_c$$

where m_c is the size of the set of tuples provided by the consumer for a given service type. The provider selects the highest ranking t_n tuples (according to the $bias$) for integration into its registry. Each provider maintains a limit (m_p) on the number of tuples it stores, which determines which of the consumer's tuples are retained.

3.3.2 Tuple Integration

There are three possible ways that each of the new tuples may be integrated into the provider's registries, based on whether the provider has existing information on α_p (specified by the tuple), namely: *add*, *substitute* or *update*. If none of the tuples in $\mathcal{S}_{type}$ refer to this provider, then the tuple is added or substituted, using the following policy: if $|\mathcal{S}_{type}| < m_p$ then the new tuple is simply *added* to $\mathcal{S}_{type}$. Otherwise, a tuple for some other provider is potentially removed to allow the new tuple to be stored. The way this is done is based on the $bias$ value; the provider inspects those tuples for which the $bias$ is less than that in the new tuple, and randomly selects one of these to be *substituted*. If none are found, then the new tuple is not introduced. This mechanism guarantees that only knowledge that has an equal (or higher) $bias$ than that existing within a provider's memory will be introduced.

The third integration mechanism, *update*, modifies an existing tuple maintained by the provider. As this new tuple represents an additional, subjective evaluation of provider α_p, the new $bias$ is calculated by averaging the new and previous tuple.

3.3.3 Sharing knowledge with Consumers

Each provider shares all the knowledge it possesses about other providers (i.e. $\mathcal{S}$), based on its aggregated knowledge shared by different consumers. As consumers will have an existing (if limited) knowledge of some service types, the shared knowledge, $\mathcal{S}$, will be organised into two distinct subsets: $\mathcal{S}^{act}$ and $\mathcal{S}^{inact}$. The set $\mathcal{S}^{act}$ contains all the tuples $r_p \in \mathcal{S}$ where the $type$ of r_p corresponds to that known by the consumer (i.e. $type$ appears in $\mathcal{R}_c$, and thus is considered *active information*). The remaining tuples are considered *inactive* information, shared (indirectly) between different consumers through common providers.

The proportion of data retained from each of these new sets is determined by the current stress of the consumer, such that the higher the value of c_s, the greater proportion of tuples from $\mathcal{S}^{inact}$ will be retained. The consumer orders each set (based on $bias$), and selects $min(0, |\mathcal{S}^{act}| - 2c_s)$ tuples from $\mathcal{S}^{act}$, and $max(2c_s, |\mathcal{S}^{inact}|)$ tuples from $\mathcal{S}^{inact}$ for retention. The top ranking t_n tuples are then integrated into the consumer's local registry, $\mathcal{R}_c$ using the same mechanism as that described in Section 3.3.2. The only difference is that the $bias$ of all $r_c \in \mathcal{S}^{inact}$ are simply replaced by $bias = 1$.

This process facilitates the discovery of new services from a group of providers that may not offer a desired service type, but

[4]The order of equally biassed tuples is arbitrary, and thus may vary whenever the set is inspected.

[5]Whilst this switching process has no explicit economic cost, it has an implicit cost as the process takes a finite time ($T_s = 2s$), during which no other service can be provided.

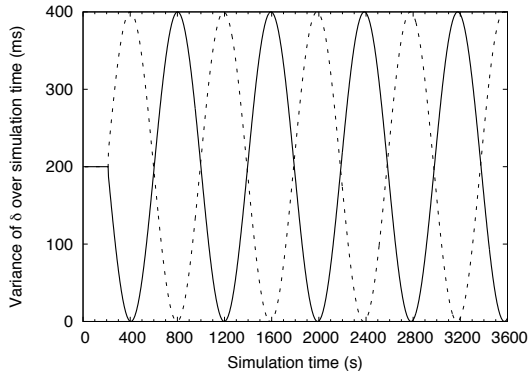

Figure 1: Variance of $\boldsymbol{\delta}$ parameter stimulating the change in demand of services, for type $\boldsymbol{A}$ (dotted line) and $\boldsymbol{B}$ (continuous line).

through consecutive interactions, may reconfigure to satisfy new consumers. Thus, consumers are willing to retain a small number of tuples pertaining to these providers within their local memory only when their allocative stress is high. As this, from a global system point of view, may indicate change in the demand occurring within the system, this mechanism allows providers to 'migrate' to a community with increased service demand. However, introduction of information into the consumer's registry does not guarantee that the provider will change its current service offering.

3.4 Global information

The model presented above limits both the quantity of information shared, and the volume of information retained by each of the agents. However, many multi-agent systems assume the use of a single middle-agent to support service discovery, such as a single *Matchmaker* or *Broker* [15], which could in theory support the task of load-balancing. In addition, the El Farol Bar problem [1] was illustrative of the characteristics encountered when global information was available.

By removing the limits on both information flow and storage capacity, the behavior arising from using a centralized service registry would be simulated, as every consumer would rapidly acquire complete and identical models of the service landscape as every other agent. Likewise, every provider would be aware of all requests from all consumers, and thus would have the same information as every other (rational) provider. This configuration provides an ideal baseline with which to compare the use of limited information flow.

4. MODEL EVALUATION

To investigate the performance of limiting memory size and information flow, and compare it to the global-information model described above, we have explored a scenario where providers dynamically adapt the type of services they offer over time in response to the changing service demands of the consumer community. Each provider may choose to offer either of the services from the service set $R = \{A, B\}$, but at any given time, a given provider can only advertise and provide a single service $s \in R$. Providers determine the type of service that they offer based on their experience of the demand for each service. Thus, when the number of providers and consumers are equal and static, service supply may converge to satisfy demand; the rate of convergence is dependent on the resource management strategies adopted by the agent community.

The consumer demand for services is varied exogenously, with the demand for service type A oscillating in anti-phase with demand for service type B (see Figure 1). This variation in demand is implemented by altering the sleep period between successive service requests. This period is defined as $c_{st} = r+\delta$, where r is a random value drawn from the range $[0, \omega]$ (where for our experiments, $\omega = 50ms$, and $T_q = T_i = 50ms$), and δ is derived from the sinusoidal function of time (illustrated in Figure 1) for each consumer requesting a service of type A or B. Thus, by varying δ for one service type (and a corresponding δ' which is 180° out of phase, for the other service type), a symmetrical change in demand can be achieved for the two services. To allow the system to achieve a steady state before dynamic demand is introduced, demand for both types of service is held constant and equal for the first $200ms$ of the simulation time[6]. To ensure that there is some knowledge about providers within the agent environment, consumers are initialized with randomized tuples.

Within each evaluation, several simulation runs were performed with different memory sizes, m_s to evaluate the model with limited information flow, and these results were contrasted with a *global* information model, which reflects the El Farol Bar problem. As the model itself assumes two separate size parameters: m_c, which corresponds to the size of the set of tuples provided by the consumer for a given service type; and m_p, the number of tuples stored by a provider for a given service type, we assume in all evaluations that $m_c = m_p = m_s$.

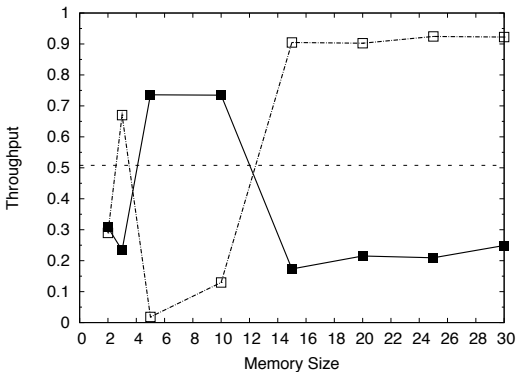

Figure 2: The mean system throughput (solid rectangles) for different memory retention sizes (i.e. varying m_s). The throughput of those consumers that update their local knowledge is represented by the line with empty rectangles. The dotted line corresponds to the case where $m_s = \infty$; i.e. the global information model.

4.1 Service management in a dynamic environment

Figure 2 illustrates the mean performance of the model (in terms of successful service allocation, or *Throughput*) with respect to the memory size, m_s. The horizontal dotted line corresponds to the case where global knowledge was available (i.e. $m_s = \infty$), and as such is a constant within this graph. The graph is normalised with respect to the optimal system performance experienced by the system in equilibrium during steady state (i.e., when service-demand is satisfied by supply such than no reconfiguration of providers is necessary).

An analysis of the model's efficiency in successfully satisfying service requests for different sizes of memory (Figure 2) reveals

[6]Whilst this scenario may not be realistic, the intent is to evaluate the behavior of the system (and resulting communities) whilst maintaining full control of the demand dynamics. More complex scenarios (and real-world case studies) could be explored in future.

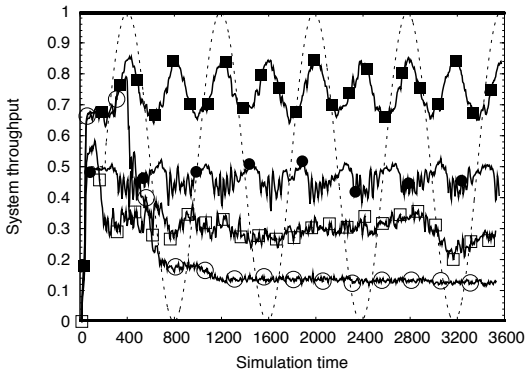

Figure 3: The performance (i.e. system throughput) of four system configurations with differing memory sizes: $m_s = 2$ (empty rectangles), $m_s = 5$ (solid rectangles), $m_s = 20$ (empty circles), and $m_s = \infty$ (solid circles).

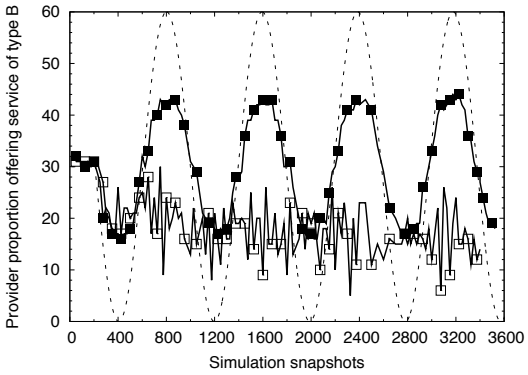

Figure 4: This shows the number of providers currently offering service of type B with respect to changing demand for B (dotted lines) for two system configurations: solid rectangles ($m_s = 5$) and empty rectangles ($m_s = 20$).

that the system efficiently satisfies service requests only for certain cases where $4 < m_s < 12$. In addition, the level of information flow between consumers is low under such conditions, suggesting that the distribution of knowledge across different consumers regarding local providers is relatively stable. However, for memory sizes outside this range, performance falls to a level below that achieved if global knowledge was available ($m_s = \infty$), where the flow of information between consumers is maximised.

Three general types of behaviour can be observed, based on the way in which the model responds to changing conditions for different memory sizes. These are captured in Figure 3, which illustrates the behaviour emerging from three representative cases (i.e. $m_s \in \{2, 5, 20\}$). When $m_s = 2$ (Figure 3 empty rectangles), the performance degrades as a result of the consumers inability to resolve service request conflicts in a timely manner. As a result, stress increases in an increasing number of consumers, which catalyses the reorganisation and sharing of the consumer's local knowledge, resulting in the acquisition knowledge about new providers which are believed to be more suitable to consumers tasks. However, given the high volume of shared information from stressed consumers, providers become selective regarding the information they retain. Since the number of retained types is so small, information loss occurs, resulting in the formation of isolated groups of agents aware of only small groups of providers, which are unable to propagate this information to other, similar groups.

The model for $m_s = 5$ exemplifies a scenario in which efficient performance is exhibited (Figure 3 solid rectangles). In this case, the model responds smoothly to changes in demand, due to the fact that the consumers' service registries are large enough to retain some information about local consumers despite the change in demand, but small enough that little of the new information exchanged is retained, and hence does not compromise future service provision. As a consequence, there are comparatively few consumers that experience stress high enough to necessitate a continual reorganisation of the local service register (as in the case when $m_s = 2$), and thus require new information (less than 10%). This also limits the number of providers that observe large changes in service demand, and thus reduces the number of providers that subsequently *switch* services. The way in which providers *switch* services in response to the changes in demand is illustrated in Figure 4, where $m_s = 5$ (solid rectangles). During the initial 200ms, local communities start to form corresponding to the two service types, with roughly equal numbers of providers supporting each community. However, as the demand for one service type increases, the level of reorganisation experienced by the consumer population is limited, and thus the number of providers that *switch* matches the resulting demand, yielding a stable, but dynamic environment.

As the number of known providers increases (e.g. in the case where $m_s = 20$), the performance becomes degenerate (empty circles in Figure 3). In this case, as consumers become stressed due to increases in service demand, they exchange and retain larger quantities of information regarding new providers. This compromises their ability to successfully locate service providers, as there will be greater competition for the highest ranking ones. This leads to an increase in the stress of the consumers, which in turn exerts unnecessary pressure on providers to *switch* due to a continuous and elevated exchange of information. Thus the equilibrium destabilises, as providers fail to respond to changes in the service demand (as illustrated by the empty rectangles in Figure 4), and the community structure is lost.

A fourth case, where $m_s = \infty$, corresponds to that where global information is available to all agents (solid circles on Figure 3). In this case, *hearding* occurs, where a large number of consumers end up competing for the most attractive providers. As these providers can only satisfy a single request at any time, a substantial number of requests are rejected, resulting in large numbers of new queries being issued to other service providers. Although consumers will eventually locate an available provider, this can take a substantial time (and hence a substantial loss in cost or utility). This results in an increase in stress experienced by a large number of consumers, which increases the pressure on providers to continually *switch*, thus further destabilising any community structure. The resulting behaviour of providers (which offer services of a given type) varies in a similar way to that observed when $m_s = 20$ (empty rectangles in Figure 4).

5. DISCUSSION

A critical factor in achieving the efficient performance demonstrated by some of the parametrisations of the model is the organisation of knowledge across the consumer and provider populations. A system where consumers successfully organise their local knowledge supports ongoing interaction between providers that can supply the services that the corresponding consumers require. Moreover, such an organisation also has the ability to smoothly reconfigure the supply of services in response to changes in demand.

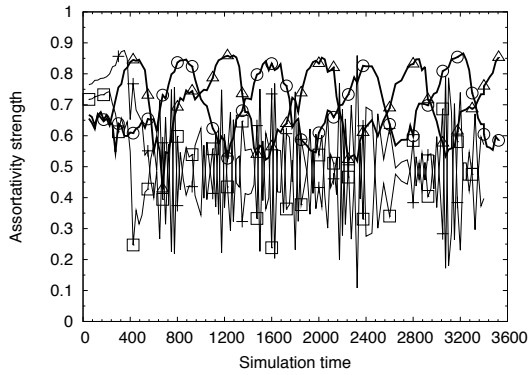

Figure 5: The assortativity strength for communities providing service types A and B over time. The triangles represents community strength for service community A (where $m_s = 5$), whereas the circles represent the corresponding community strength for community B. The rectangles and crosses denote the corresponding populations (for A and B) when $(m_s = 20)$.

In contrast, operating on the wrong information will result in degraded performance due to high numbers of rejected queries and time spent on needlessly reconfiguring providers to change their service offerings. Populations of agents that possess only incomplete local knowledge must therefore rely on the appropriate flow of information in order to keep this knowledge up to date, and coherent enough to be of use. This must be achieved in the absence of any global flow-facilitating mechanisms determining *what* information should flow to *which* agent.

To analyse the structure of the information that individual agents utilise when either provisioning services (in the case of consumers) or in deciding whether to reconfigure their service offerings (i.e. for providers), an *assortativity metric* has been defined, which represents a measure of the community strength based on the information shared between a population of agents $\mathcal{A}_{type}$ interested in (or offering) a given service type $type$. For each population, a square matrix $M(\mathcal{A}_{type})$, is generated of size $n_{type} \times n_{type}$ (where n_{type} corresponds to the number of providers), and each element represents the number of times a pair of providers α_p^i and α_p^j are known by the same consumer α_c^k. The higher the value, the more aware the consumers are of both providers, and hence the likelihood of both providers being in the same community is greater. This matrix can then be realised as a network in which the nodes represent different providers, and the edge represents an association between the two providers, reflecting the fact that they are both known by the same consumer. The *strength* of the community structure represented by such a network can be calculated by considering the types of service offered by the providers which are linked. Thus, for a provider offering a service of type $type$, the assortativity metric (H_{type}) is defined as follows:

$$H_{type} = \frac{1}{n_{type}} \sum_{\alpha_p^i}^{n_{type}} d_{\alpha_p^i}$$

where $d_{\alpha_p^i}$ is the proportion of neighbours of α_p^i that offer service $type$, and n_{type} is the number of providers of service type $type$ with at least one neighbour.

The results of this analysis are illustrated in Figure 5. Here, the manner in which service assortativity varies over simulation time for a system operating on $m_s = 5$ (triangles for providers of service A and circles for providers of service B) and $m_s = 20$ (rectangles for providers of service A and crosses for providers of service B) is shown. A system that effectively manages service allocation (circles) is characterised by a community that is strong enough to sustain a stable system reorganization. As the consumers experience increased stress due to rising demand for a given service type, they exert increasing pressure on the providers to satisfy this demand, which in turn strengthens the community structure. For the small memory model (i.e. when $m_s = 5$), the community strength approached 0.9 as the demand for a service type reached its peak; reflecting the fact that the majority of knowledge retained by the consumers was correctly identifying relevant providers. Correspondingly, as the service demand fell, the strength of the community decreased accordingly. The larger memory model (i.e. when $m_s = 20$) illustrates the pathological case, whereby providers *switch* far too frequently due to the retention of too much knowledge, compromising the ability of consumers to effectively locate relevant providers. In this case, the community strength becomes erratic, and no longer varies in phase with changes in service demand.

These results suggest that the formation of strong and adaptive communities is accompanied by high system performance and a robust response to the varying demand. Recall that service provision depends on the co-adaptive stability between agents responding to locally perceived changes. The formation of a strong community by agents interested in a particular service type yields an exchange of information that is *constrained*. If a community is strong, knowledge passed to the consumers can be exploited to induce a small subset of providers to *swich* service types, minimising the risk of destabilising the availability of service to other consumers in other local communities. This raises the question of how strong should a community be in order to support this process optimally? A tightly linked community (where community strength approaches one) could prevent reorganisation when there are changes in demand. However, if the community is too weak, then the availability of services will be destabilised. A second question thus arises: under what conditions is such a system capable of forming strong communities? As demonstrated in this paper, simply varying the amount of information available to agents is sufficient to bring about changes in system behaviour: from being too disconnected, to effectively evolving self-organising communities, and eventually to a configuration where agents regress into a chaotic flux.

Figure 6 plots the community strength as a function of memory size. In those cases where the size is too small or too big, degenerate behaviour results, similar to that illustrated for the large memory model ($m_s = 20$) in Figure 5. For those cases where $m_s \in \{2, 3\}$, consumers retain insufficient knowledge to successfully facilitate a useful flow of information as the environment changes. In contrast, when the memory size is large (i.e. $15 \leq m_s$), consumers retain more information about the environment, and consequently can better stimulate change in the provider population. As levels of stress increase, there is a greater chance that larger numbers of consumers will compete for the most favoured providers (i.e. herding), resulting in a feedback loop that escalates stress levels, which increases the pressure on providers to change, yielding further destabilisation. It is only within the small range of cases (i.e. $4 \leq m_s < 15$), where the community strength increases. Here, each agent is able to adapt to changes in the environment effectively, without necessarily being able to identify the globally optimal providers. The close agreement between Figures 6 and 2 demonstrates that it is the strength of the communities formed through information flowing through the system that directly underpins and accounts for system performance.

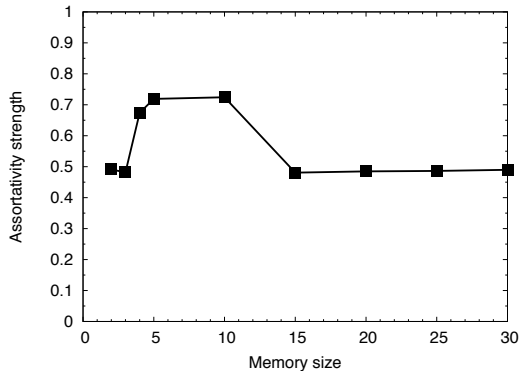

Figure 6: The mean assortativity strength as a function of memory size.

An analysis of the information flow that emerges from the evaluation of our model can provide a number of insights into the dynamics of self-organisation, and in particular, how self-organisation can emerge from localised decision-making within the context of resource management problems involving configuration of providers to offer services based on the locally perceived demand. Although the results demonstrate that the size of the local registry containing provider information is one of the critical factors influencing the flow of information between agents in the neighborhood, and thus the cohesion and stability of the community as a whole, there are other interdependent variables that play an important role in the process. Such variables include: the frequency of information exchange, the quantity of information exchanged in a single exchange transaction, and thus the level of influence an agent has on another agent's decision making. These variables have been demonstrated to play a key role in achieving self-organisation and adaptation for a number of other decentralised architectures that have been investigated [5, 3, 7]. In the model presented in this paper, the size of the exchanged information was dynamically affected by consumer *stress* (c_s) that determined the amount of information that both consumer and provider agents were willing to accept. Likewise, the level of consumer stress was an important factor in determining the impact of the consumer, whilst querying the provider for a given service type. Finally, this stress level was also significant in establishing the demand for a scarce resource, and hence encouraging providers to switch the type of service they currently provided. As all of these factors have an impact on how agent interactions evolve, and thus how information flows through the system, we plan to conduct a further investigation to analyze how these parameters affect each other in order to formally identify the synergistic relationship between these parameters, and to identify efficient techniques for automatically *self-regulating* them in a decentralised manner based on information available locally to each agent.

6. CONCLUSIONS

In this paper, we have investigated how adaptive service provisioning can arise out of local interactions between service providing and consuming agents. Using an adaptive multi-agent system model, a relationship between the level of information exchanged and the size of emergent agent neighborhoods was identified, as well as the systems ability to self-organise, to form stable, local communities. The results obtained suggest that when information exchanged by agents is communicated only to a small subset of their local peers, the system has the capability to self-organise into communities within which service providers reliably provide the most requested service types, and consumers are better able to successfully provision services due to locally propagated information regarding known providers. This stability is maintained through a pertinent information flow that keeps community members aware of both recent service demand, and service availability.

7. ACKNOWLEDGEMENTS

The work was partially funded by the Engineering and Physical Science Research Council's AMORPH grant EP/D00232X/1.

8. REFERENCES

[1] W. B. Arthur. Inductive reasoning and bounded rationality. *American Economic Review*, 84:406–411, 1994.

[2] S. Brueckner and H. V. D. Parunak. Self-organizing MANET management. In G. D. Marzo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, pages 1–16. Springer, 2003.

[3] S. A. Brueckner and H. V. D. Parunak. Information-driven phase changes in multi-agent coordination. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 950–951. ACM Press, 2003.

[4] E. H. Durfee. Scaling up agent coordination strategies. *IEEE Computer*, 34(7):1–8, 2001.

[5] S. Guerin and D. Kunkle. Emergence of constraint in self-organizing systems. *Journal of Nonlinear Dynamics, Psychology, and Life Sciences*, 8, 2004.

[6] T. Hogg and B. A. Huberman. Controlling chaos in distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 21:1325–1332, 1991.

[7] N. H. Packard. Adaptation toward the edge of chaos. In A. M. J.A. Kelso and M. Shlesinger, editors, *Dynamic patterns in complex systems*, pages 293–301. World Scientific, 1988.

[8] H. V. D. Parunak and S. A. Brueckner. Engineering swarming systems. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, pages 341–376. Kluwer, 2004.

[9] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[10] S. Sen, S. Roychowdhury, and N. Arora. Effects of local information on group behavior. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 315–321. AAAI Press, Menlo Park, CA, 1996.

[11] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[12] S. Stein, N. R. Jennings, and T. R. Payne. Flexible service provisioning with advance agreements. In *Seventh International Conference on Autonomous Agents and Multi-Agent Systems*, pages 249–256, May 2008.

[13] S. Stepney. Critical critical systems. In S. S. Ali E. Abdallah, Peter Ryan, editor, *Formal Aspects of Security: FASec*, volume 2629 of *LNCS*. Springer, 2003.

[14] P. Stone and M. Veloso. Layered learning and flexible teamwork in robocup simulation agents. *Lecture Notes In Computer Science*, 1856:495 – 508, 2000.

[15] K. Sycara, K. Decker, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, January 1997.