Global Order From a Minimal Local Resource Allocation Strategy

Mariusz Jacyno Seth Bullock Terry Payne Michael Luck

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK
Email: {mj04r,sgb,trp}@ecs.soton.ac.uk, michael.luck@kcl.ac.uk

Abstract: In this paper we explore the relationship between local and global behaviour in a simple model of utility computing infrastructure as the system heterogeneity, load and reliability are varied. To do this, we implement minimally complex agent strategies for which we can identify the fundamental generic feedback underlying system behaviour. Such feedback must be balanced by any utility computing infrastructure if decentralised control is to become an effective technique for preserving stable functionality.

Keywords: decentralised control, multi-agent system, self-organisation, decentralised resource allocation

1. Introduction

The advent of the Internet and distributed computing has enabled remote access to a range of computational resources such as CPU power, disk storage, bandwidth, laboratory equipment, etc. This ability to share and utilise computational resources across a group of independent users has created an opportunity to both simplify the traditional service provisioning model and lower service management and provisioning costs.

In response to these possibilities, a number of initiatives, such as *autonomic computing* [6] and *utility computing* [9], have been announced by major IT vendors sharing the same underlying principles of provisioning resources on demand to a large number of users. Since these infrastructures are large, open and dynamic systems that are free to change and grow organically by introducing or removing new components in an *ad hoc* manner, control over their resource allocation presents unique challenges that may overwhelm existing centralised management approaches [2].

Designing effective control mechanisms capable of simultaneously delivering both low cost of overall infrastructure operation, and cheap service provision for users, requires a solution that is scalable, reliable and adaptive to changing system conditions. In this context, the application of a multi-agent system involving a population of active and autonomous agents offers one way to address the challenge, but it is far from obvious how the individual system elements need to be designed to meet these objectives. In particular, this approach addresses the problem of designing decentralised resource allocation mechanisms which, despite operating on local and imperfect information, lead to the satisfaction of global system objectives.

Rather than attempting to assess candidate solutions to the resource allocation problem, or optimise system performance in some way, in this paper we focus on understanding the fundamental nature of the interactions between agents arising from their individual local control mechanisms. Unfortunately, characterising the manner in which particular resource allocation mechanisms bring about such feedback in full-blown systems can be extremely challenging. Here, we focus on understanding a system featuring minimally complex resource allocation mechanisms, where such a characterisation is more tractable. In doing so, we are motivated by the working hypothesis that the dynamics of such a system, while simple, will share general properties with more realistic utility computing systems, and a belief that achieving an understanding of decentralised control at this level of abstraction will be crucial if it is to become an effective means of managing distributed open systems.

2. Prior Work

The introduction of shared and constrained resources into systems of the kind described above may quickly and unexpectedly lead to the emergence of undesirable system behaviour, often described as *resource competition* [7]. Here, consumers end up competing for specific subsets of resources, leaving others under-utilised. The resulting poor global resource utilisation may eventually cause unstable and unpredictable system functioning. Furthermore, since there is no centralised control, the system may reinforce this competitive behaviour leading to a very rapid degradation of system performance [5].

Intelligent resource allocation in systems of this kind have been approached via work on coalition formation [12], group problem solving [13] and teamwork [8]. A common property of such models is their reliance on distributed protocols and focus on design of intelligent algorithms coordinating the behaviour of agents. However, these mechanisms, whilst decentralised, generally assume up-to-date shared information, and thus in order to converge to an optimal solution, they require a substantial amount of global system information followed by a large number of interactions among system elements to maintain awareness of peer goals, actions, etc. As a consequence, they are often vulnerable to increasing system scale and/or dynamism.

A smaller number of studies focus on systems where agents do not have access to global information and thus operate on simple and local algorithms [11, 4, 1]. Despite these limitations, such architectures can be robust to failures and may exhibit interesting self-organising properties that are difficult to explain considering the minimal intelligence of the individual elements.

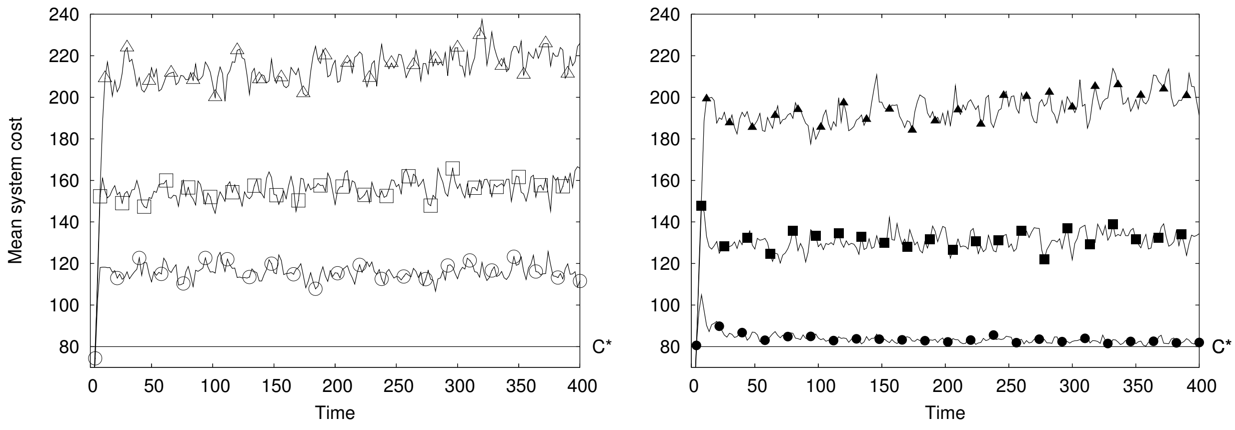

Fig. 1. Relation between mean system cost and system load for agents relying on $\mathcal{R}$ (left) and $\mathcal{RP}$ (right). Three levels of system load are represented: $L = 1$ (circle), $L = 2$ (box), $L = 3$ (rectangle). The dotted line (C^*) corresponds to optimal system cost. In each case, $S_N = 240$, while U_N is varied from 120 through 360.

The approach adopted in our work shares the same motivation of understanding how global system stability can arise when agents perform resource allocation independently, i.e., without centralised executive control. However, whereas previous work has investigated how the heterogeneity of the system can lead to stability through either limiting the knowledge possessed by individual agents [11] or by designing local decision procedures that diversify agent behaviour [4], here we focus on identifying general principles that explain how performance scales under various kinds of systemic pressure. As such, we do not aim to develop a working solution to a specific resource allocation problem for utility computing. Instead, we concentrate on characterising relationships between local mechanisms and global performance that are likely to generalise from the minimal system explored here to real-world utility computing systems in general.

More specifically, we are interested in explaining the way in which different consumer agent strategies give rise to both a certain level of *efficiency* and a particular degree of *fairness*. This requires us to analyse not only global behaviour but also the performance of groups of similar agents, to determine how the costs associated with resource allocation in a utility computing infrastructure are distributed across a population of heterogeneous agents.

3. Simulation

The decentralised multi-agent system that we will explore in this paper comprises of a service registry that serves as an inventory of the resource providers within the system; a population of S_N agents representing resource providers (services); a population of U_N agents representing resource consumers (consumers).

Services are provided by agents that facilitate access to resources (disk storage, CPU time, etc.).

Consumers consume resources according to a fixed personal workflow defining the type, capacity and order of services required.

The registry is an agent tasked with maintaining an inventory of system services, and supplying it to consumers when queried. Since, in reality, the information obtained from such registries can become stale and unreliable in a dynamic system, in future work we will be interested in systems where this unreliability is sufficient to ensure that consumers prefer not to make use of it at all.

Service Allocation is decentralised such that each consumer first obtains from the registry a list of existing services capable of providing any of the resources required by its workflow. This action takes time T_x and incurs an execution cost, C_x. Repeatedly, services are chosen from this list and their availability determined (each time incurring a query cost, C_q, and consuming time, T_q). Services may be unavailable because they are busy to the extent that they do not have the spare capacity required by the workflow component, or because they are no longer part of the system. Once an available service has been located, the agent attempts to allocate the next component of its workflow. Once all components of the workflow are allocated to services in this way, the agent attempts to execute the workflow using these services. Since services are not locked during the allocation process, it is possible that a consumer agent may allocate a workflow component but find that the service is busy when it attempts to execute it. In such circumstances, the consumer must re-allocate this workflow component. Successfully executing a component also takes time, T_x, and incurs an execution cost, C_x. Should a service fail during execution, the consumer still pays the execution cost, but must also re-allocate the workflow component. If, during any allocation process, a consumer makes n attempts to locate an available service of a particular type, the allocation is deemed to have failed, as is the workflow it is part of. Here, for each workflow component to be allocated, we set n equal to the number of services of the required type returned by the registry. Whether successful or unsuccessful, upon completing a workflow, a consumer agent, U_i is inactive for some randomly determined period drawn from a uniform distribution $[0, \omega_i]$ after which the same workflow allocation process begins again.

Scenarios, unless stated otherwise, involve an equal number of agents repeatedly attempting to execute each of three different workflows ($W_1 = \{A\}$, $W_2 = \{A, B\}$, $W_3 = \{A, B, C\}$) for a period of time, T_N. Since we expect simple workflows

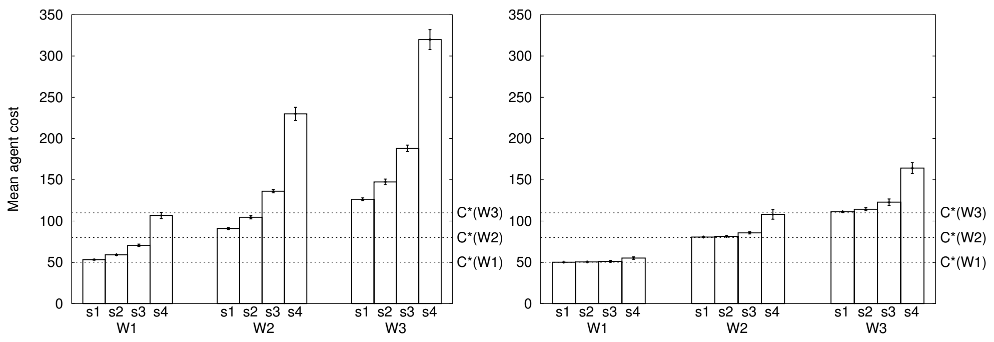

Fig. 2. Mean workflow completion costs for agents relying on $\mathcal{R}$ (left) and $\mathcal{RP}$ (right) where $H = 4$. Within each workflow type, four subclasses are identified in order of increasing capacity requirement ($s1, s2, s3, s4$). Dotted lines correspond to the optimal cost for each workflow group. $U_N = 180$, $S_N = 360$, $T_N = 400$ seconds.

to be requested more frequently, the maximum period of time for which a consumer will sleep after completing (or failing to complete) a workflow, ω, is workflow-dependent such that more complicated workflows tend to be associated with longer periods of sleep: $\omega_1 = 1$s, $\omega_2 = 2$s, $\omega_3 = 3$s.

For all agents, the following values are assigned to the costs and execution times of service interactions and service executions: $C_x = 20$ units, $T_x = 500$ms, $C_q = 10$, $T_q = 100$ms. These parameters define minimal costs for a consumer attempting to execute each workflow: $C^*(W_1) = 50$ units, $C^*(W_2) = 80$ units, $C^*(W_3) = 110$ units. Since the proportion of consumers attempting to execute each workflow is the same, the optimal cost for a system can be calculated as $C^* = 80$ units.

Consumers rely on strategies to guide their individual behaviour. Here we explore minimally sophisticated strategies: **Random selection** ($\mathcal{R}$): when attempting to allocate a workflow component to a service, the consumer makes random choices from the list of available resources obtained from the service registry. **Hybrid strategy** ($\mathcal{RP}$): when attempting to allocate a workflow component to a service, the consumer preferentially returns to the last service employed for that component, if it was executed successfully during the last workflow, otherwise the random selection strategy is employed.

We do not expect these simple strategies to be employed within real utility computing systems. However, the simplicity of the randomising and canalising behaviours that they employ make them good candidates for examination, since any decentralised resource allocation strategies that *are* employed within utility computing will probably involve some more complicated form of randomisation and/or canalisation.

4. Results

Here we characterise the behaviour of the minimal simulated system, concentrating on the manner in which system performance scales with four system parameters: *heterogeneity*, *load* and *reliability*. In each case, we are interested in both the efficiency of the system as a whole, and the efficiency of the consumer agents within it. The former is measured over a specific test period by calculating the average cost per executed workflow. This measure makes sense where consumers never (or rarely) fail to execute a workflow. At the level of individual consumers, we are interested in any advantage that one class of consumers (say those attempting to execute a simple workflow) might have over another. In each case, we are interested in how different consumer agent strategies impact on these measures.

4.1 System Load

Since real utility computing infrastructures must cope with variation in both the level and type of demand for (and provision of) services, demand may sometimes outstrip supply, precluding optimal allocation. To investigate the impact on system behaviour of variation in the balance between supply and demand, we vary the system load, L, defined as the ratio of consumer demand to service provision. For a system where $L = 1$, in principle every workflow component can be simultaneously satisfied by system services. Doubling this load ($L = 2$) ensures that only half of the consumers' workflow components can be executed simultaneously. Here, system load is manipulated by holding the number, type and capacity of system services constant, and varying the number of consumers (but not the proportions of different workflows being allocated). Doubling the number of consumers thus doubles system load. Scenarios are identical to those described in Section 3 save that there is heterogeneity in the demand for capacity across three different workflows ($W_1 = \{A_{10}\}$, $W_2 = \{A_{11}, B_{11}\}$, $W_3 = \{A_{12}, B_{12}, C_{12}\}$).

How does the system respond to increasing load? Figure 1 illustrates the relationship between mean system cost and load for the two strategies. The ability of $\mathcal{RP}$ to approach optimal performance where supply matches demand ($L = 1$) is lost for higher system load, and the advantage it enjoys over $\mathcal{R}$ is reduced. Neither strategy can cope with the increased number of "collisions" during resource allocation that result when consumers can no longer utilise preferred services exclusively.

4.2 System Heterogeneity

It is highly unrealistic to assume that agents attempting to allocate the same type of resource will also share exactly the same service preferences. For example, in the domain of utility computing, different amounts of CPU processing power, storage size, quality of service, etc., may be required. For each consumer, only a subset of services of a particular type will be capable of satisfying its particular demands. Since many of these attributes are dynamic properties that may change rapidly and unpredictably, it may be that a centrally maintained registry of services cannot be relied upon to provide the information required by consumers to identify appropriate services.

In order to manipulate the degree of heterogeneity in consumer demand, H, within the model we assign different capacity requirements to consumers and differing capacity provision to services. A consumer will be satisfied by any service of the required type with free capacity that either equals or exceeds its capacity requirements. As such, consumers with high capacity demands must necessarily have at least as difficult an allocation task as consumers with lower capacity demands. In all cases considered here, there exists an allocation of services to consumers where all available service capacity is utilised in executing every workflow component simultaneously (i.e., supply always exactly matches demand), and no service has capacity to simultaneously execute more than one workflow component. We define H as the number of unique levels of service capacity required by the workflows of a consumer population (or, equivalently, the number of unique levels of capacity provided by a population of services). Thus, for $H = 1$, all workflows share the same capacity requirements, whereas for $H = 2$, each workflow (and every service type) is present in a low-capacity and high-capacity variant such that each variant is assigned to an equal number of consumers. The scenarios reported below are otherwise identical to those described in the *Simulation* section above, with systems comprising 180 consumers and 360 services respectively.

Figure 2 illustrates the mean workflow completion costs for consumers relying on the two most successful strategies, for the highest degree of consumer heterogeneity ($H = 4$). Four levels of capacity demand are depicted for each workflow $\{s1, s2, s3, s4\}$. While high-capacity workflows tend to attract higher allocation costs, irrespective of consumer strategy, the departure from optimal allocation costs is much reduced for $\mathcal{RP}$ strategists. Consumers adopting this hybrid strategy were thus able to form preferences for resources that not only satisfied their own resource demands but, in the case of low-capacity consumers, also contributed to satisfying the demands of their competitors, increasing overall system efficiency.

Convergence to an optimal allocation is a consequence of a balance between random selection and preferential selection. Where an inefficient allocation of services to consumers arises, the former tends to disturb preferences for resources that are overexploited, whilst preferential selection is able to increase pressure on under-exploited services. In this way, the extent to which low-capacity consumers form preferences for high-capacity resources will tend to be matched by the pressure to "move on" exerted by unsatisfied high-capacity consumers. Existence of this pressure is visible in Figure 3, which illustrates the mean system cost for increasing degrees of heterogeneity. Here, unsatisfied high-capacity consumers initially exert strong pressure on preference reconfiguration, enabling the system to converge to a more optimal state.

4.3 Service Reliability

A further distinguishing characteristic of utility computing infrastructures is the lack of assurance that existing services will not fail or become unavailable during a consumer's lifetime. To investigate the impact of resource failure, randomly selected services are removed from the system at a constant rate. The scenario is initially identical to that described in Section 4.1, with 360 unallocated services in principle exactly matching the demand of 180 consumers. However, after a 40-second period of normal service allocation during which time the system settles to its typical behaviour, services begin to be removed at random at a rate of one per second, until none remain.

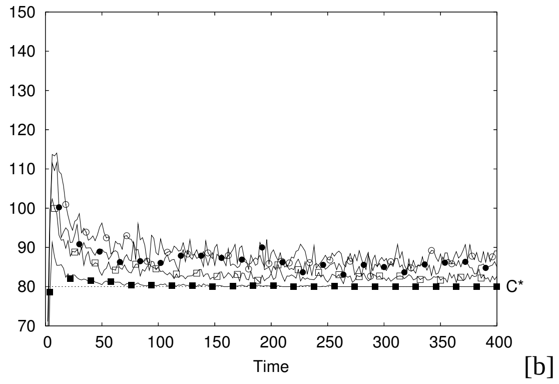

Fig. 3. Relation between mean system cost and consumer heterogeneity for agents relying on $\mathcal{RP}$ strategy. Four degrees of consumer heterogeneity are depicted: $H = 1$ (solid rectangle), $H = 2$ (empty rectangle), $H = 3$ (solid circle) and $H = 4$ (empty circle). The dotted line C^* corresponds to optimal system cost. $U_N = 180$, $S_N = 360$.

Figure 4 illustrates the manner in which mean system cost varies over time in such a scenario, both for $\mathcal{R}$ strategists and $\mathcal{RP}$ strategists. Over the majority of the simulated period, the $\mathcal{RP}$ population enjoys an advantage over the $\mathcal{R}$ population in terms of allocative efficiency. However, this advantage decreases over time. For each population, costs rise with increasing service failure at an accelerating rate, until a catastrophe is reached at around 360 seconds. At this point, certain types of resource are no longer present within the system, preventing some workflows from being completed successfully. By 400 seconds, all consumers are paying a cost associated with accessing the registry, but are failing to carry out any allocation.

For the scenarios simulated here, over time, as services fail, system load increases. It is instructive to compare the mean system cost that results from service failure to that reported for the same constant system load. Prior to the first resource failure at $t = 40$, system load has been stable at $L = 1$. Subsequently, as resource failure increases system load, it is remarkable that consumers are able to achieve an allocative efficiency equivalent to a system under constant load for load values as high as $L = 12$. Despite the scale of the system and the simplicity of the resource allocation mechanisms, it is evident that the allocative reconfiguration required by increasing load can be achieved smoothly and efficiently.

As noted above, as the number of failed resources (and thus system load) increases, the advantage that $\mathcal{RP}$ has over $\mathcal{R}$ in terms of allocative efficiency diminishes until, under extreme system load, it disappears. This result can be interpreted as indicating that the role of preferential selection within the $\mathcal{RP}$ strategy also diminishes over time, with behaviour increasingly

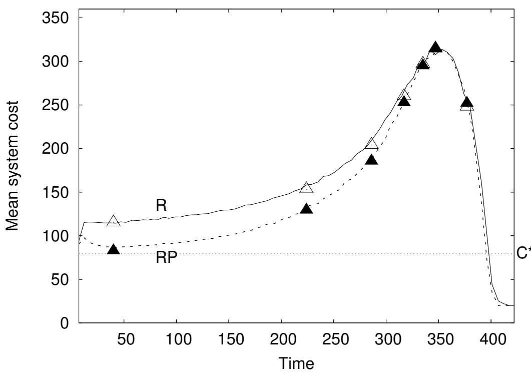

Fig. 4. Relation between mean system cost and degree of resource failure for consumers relying on $\mathcal{R}$ (solid line) and $\mathcal{RP}$ (dotted line). Initially, $U_N = 180$, $S_N = 360$. From the $40^{\rm th}$ second, one randomly selected resource fails permanently each second. Symbols indicate the mean system cost experienced at an equivalent constant load, calculated over a window $300{\rm s} < t < 400{\rm s}$, for load values drawn from $\{1, 2, 3, 4, 5, 6, 12\}$.

dominated by random selections at high system load. This interaction is discussed further, below

5. Discussion

Like any open system, a utility computing infrastructure must operate under a certain amount of stress or pressure. This paper represents a first attempt to understand *how* such a system responds to this pressure, and *why*. In general, open systems have a tendency to self-organise in response to systemic pressure. In the Bernard experiment [14], for instance, driving up the temperature of a fluid encourages the system to structure itself such that its heat is dissipated more effectively—an opportunistic equilibrating response that is in opposition to the pressure that triggers it. Furthermore, the further such a system is perturbed from equilibrium, the more sophisticated are its mechanisms for resisting the perturbation [10].

Here, at any point in time a population of $\mathcal{RP}$ consumers can be represented as two interdependent sub-populations of agents: one currently able to rely on a developed preference, while the other either has no such preference, or has a preference for a service that is currently busy, and must rely on random selection. While it is important to remember that every consumer in a $\mathcal{RP}$ population possesses the same strategy, we will refer to these temporary behavioural dispositions as $\mathcal{RP}$ strategy *elements*. Agents relying on the $\mathcal{R}$-element are more aggressive, selecting resources randomly and thereby dispersing their activity across all system resources. Agents relying on the $\mathcal{P}$-element, on the other hand, canalise their activity in a specific region of the systems resources. Where supply meets or exceeds demand, the former encourages system fairness, while the latter lowers system cost.

Since there is no central controller deciding which agent should rely on which strategy element, it is interesting to explore by what means the balance between strategy elements is brought about. Crudely, each "sub-population" exerts a specific pressure on the other. By "stealing" the preferred resources of conservative $\mathcal{P}$-element agents, aggressive $\mathcal{R}$-element agents drive $\mathcal{P}$-element agents to switch strategy. At the same time, $\mathcal{R}$-element agents that successfully allocate resources also switch strategy element. In both cases, such switching prevents agents relying upon the same resource for a long time. This ensures that the costs of resource competition are distributed fairly among all agents. Furthermore, as system dynamism increases (with increasing load or heterogeneity, for instance), and the chance of developing useful preferences falls, the proportion of $\mathcal{R}$-element agents increases. Likewise, if system dynamism relaxes, the proportion of agents successfully exploiting preferences increases. This coupling between strategy elements drives the system behaviour, and its response to externalities such as load or heterogeneity. The nature of this coupling resembles certain accounts of self-organisation within natural decentralised systems, where complex system-level dynamics are characterised in terms of the generic feedback that arise from local interactions between components [3, 7]. Within the minimal system presented here, the $\mathcal{R}$ strategy (or element) generates destablising positive feedback within the population of agents, whereas the $\mathcal{P}$-element induces canalising negative feedback.

Our results demonstrate that the system tends to adaptively balance the elements of the hybrid strategy in response to particular levels of demand. In some sense, the strategy also ensures that this balance is not achieved at the expense of fairness within the system, since, on average, every agent with the same workflow spends the same amount of time employing each strategy element. This is clearly visible in the case of heterogeneity tests, where feedback between populations of agents relying on either the $\mathcal{R}$-element or $\mathcal{P}$-element exert pressure on each other. This feedback eventually forces the reorganisation of agent preferences such that the pressure due to heterogeneity, like the temperature in a Barnards cell, is "dissipated" automatically and naturally without recourse to intelligent planning or centralised administrative intervention.

References

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

[2] S Bullock and D Cliff, Complexity and emergent behaviour in ICT systems, Technical Report HP-2004-187, Hewlett-Packard Labs, 2004.

[3] F Heylighen and C Joslyn, Cybernetics and second-order cybernetics. Encyclopedia of Physical Science and Technology 2001, R Meyers, Ed., Vol. 4, Academic Press, New York, pp. 155–170.

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

[5] T Hogg and B A Huberman, Dynamics of large computational ecosystems, Tech. Rep. HPL-2002-77, Information Dynamics Laboratory, Hewlett-Packard Laboratories, Palo Alto, 2002.

[6] J O Kephart and D M Chess, The vision of autonomic computing. IEE Computer, Vol. 36, No. 1, 2003, pp. 41–50.

[7] H V D Parunak and S A Brueckner, Analyzing stigmergic learning for self-organizing mobile ad-hoc networks (manets). Engineering Self-Organising Systems 2004.

[8] D Pynadath and M Tambe, The communicative multiagent team decision problem: analyzing teamwork theories and models.

[9] M A Rappa, The utility business model and the future of computing services. IBM Systems Journal, Vol. 43, No. 1, 2003, pp. 32–42.

[10] E D Schneider and J J Kay, Order from disorder: The thermodynamics of complexity in biology, In What Is Life: The Next Fifty Years. Reflections on the Future of Biology, M P Murphy and L O'Neill, Eds. Cambridge University Press, 1995, pp. 161–172.

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

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

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

[14] R Swenson and M Turvey, Thermodynamic reasons for perception-action cycles. Ecological Psychology, Vol. 3, No. 4, 1991, pp. 317–348.