On the Effects of Omitting Information Exchange Between Autonomous Resource Management Agents

Siri Fagernes[1] and Alva L. Couch[2]

[1]Faculty of Engineering
Oslo and Akershus University College of Applied Sciences
Oslo, Norway
siri.fagernes@hioa.no

[2]Computer Science Department
Tufts University
Medford, MA, USA
couch@cs.tufts.edu

Abstract. We study the problem of information exchange between coordinated autonomous resource management agents. We limit information to that which can be directly observed by each agent. While exchanging all relevant observables leads to near optimal management, leaving out information leads to "hidden variable" problems that affect only part of the overall behavior. The patterns observed for hidden variables in simulation predict what will happen in a realistic situation when not all information is interchanged. Through simulation, we observe that leaving out information results in non-optimal behavior of the resource management model when resource needs are decreasing, although the partial information model performs very well when resource needs are increasing.

Key words: self-organization, resource management, autonomic computing, agent management, distributed agents

1 Introduction

Traditional approaches to autonomic management of larger distributed systems involves monitoring many components, and gathering information at a global level. Many solutions also involve creating representations of the system with near-complete knowledge of both entities and their interactions. The gathering of such knowledge is costly and generates significant network overhead.

To achieve dynamic resource allocation in cloud computing, there has been an increased attention to the so-called *elasticity* of cloud data centres [1], [2]. The primary goal is to run cloud data centres cost-efficiently; to fulfil SLAs without overprovisioning. This requires a system for dynamic resource allocation, which is able to adapt to varying demands over time.

In our research we seek simpler solutions to autonomic management. Our primary goals have included decreasing information exchange among entities and decreasing the need for prior knowledge and learning. Previously we have demonstrated that timing of events in an autonomic system is crucial in to achieve efficient management [3], [4], [5]. In this study, our goal is to determine what type of information is most important to achieve sufficient management results, and whether certain types of information exchange are unnecessary.

2 Related work

The traditional approach to achieving autonomic management is based on control theory. It is based on control loops which monitor and give feedback to the managed system, in addition to making changes to the system based on the feedback. The control-theoretical approach is suited for managing closed systems, which are usually less vulnerable to unpredictable events and external forces influencing the system. It is not as successful in representing open systems, where we do not necessarily know the inner structure and relationships [6].

The control-theoretical approach involves the use of one or more autonomic *controllers*, which sense and gather information from the environment where they reside. If any global knowledge needs to be shared among the controllers, this is normally done through a *knowledge plane* (KP) [7], [8], [9]. A KP should provide the system with knowledge about its goals and current states, and hence be responsible for gathering all necessary information and also generating new knowledge and responses. This approach involves much coordination and information overhead among the networked entities in the system being monitored.

To achieve autonomic resource management based upon the above approaches, one normally uses *adaptive middleware*, which is placed between the application and the infrastructure [10], [11], [12]. This middleware mediates between managed services and clients, and reconfigures services as needed to adapt to changing needs and contingencies.

Cloud elasticity is defined as the ability of the infrastructure to rapidly change the amount of resources allocated to a service to meet varying demands on the service while enforcing SLAs [13]. The goal is to ensure the fulfilment of the SLAs with the least amount of overprovisioning. A common approach is to build controllers based on predictions of future load [13]. [1] proposes system integrating cost-awareness and elasticity mechanisms like replication and migration. The system optimizes cost versus resource demand using integer linear programming. [13] models a cloud service using queueing theory and designs a closed system consisting of two adaptive proactive controllers to control the QoS of a service. Predictions on future load is used as a basis for estimating the optimal resource provisioning.

In this paper, we study an approach to elasticity based upon autonomous, distributed agents. This differs from the middleware approach in that the agents are autonomous and distributed, and do not mediate between clients and services; they simply observe what is happening and adapt the service accordingly.

We avoid the use of a centralized planner, to increase both potential scalability and robustness, and seek instead to define autonomous, independent agents whose minimal interactions accomplish management.

3 Models and variations

The work presented in this paper is a continuance of the work presented in [3], [4] and [5], which again is based on the work presented in [14] and [15]. The original closure model [14] consists of a single closure operator Q controlling a resource variable R. The resource level determines the performance of the system, which is determined based on the response time of the service the system delivers. Decisions on resource adjustments (increase/decrease) are made based on iterative feedback on the perceived *value* of the service. Initial studies [14], [15] showed that a simple management scheme with minimal available information could achieve close-to-optimal performance. In [3], [4] and [5] we extended the original model to a two-operator model. The aim with this research was to investigate the efficiency of management when two resource variables in the same system need to be updated without access to full system information.

3.1 Single closure model

The single closure model represents a system that delivers a service S. The resource usage is modelled by a resource variable R, which in this scenario represents a number of virtual servers, and has an associated cost C. The system or service performance is measured by the response time P, which is affected by the system load L. The total value of the service is V. The system load L is defined as an arrival rate of requests, and the system performance P is defined as the *request completion rate*. The system dynamics are as follows:

- *Cost* increases as R increases. A linear relationship between C and R is plausible, i.e. $C = \alpha R$.
- Performance P increases as R increases, and decreases as the load L increases. The system performance P (in requests handled per second, a rate) has a baseline performance B (the quiescent request completion rate, or the performance when there is no load affecting the system). B is a constant value. A plausible estimate of system performance P is then the baseline performance minus corrections for load and resource usage, $P = B - \frac{L}{R}$. This has the rough shape of a realistic performance curve, though realistic curves are described by much more complex equations. The definition of P is a statement that as L increases, performance decreases; B is the baseline performance for no load. The model is an ideal case. In real situations, there would be a baseline in which B is not affected; for certain levels of L, P would be flat.
- A plausible concept of value is βP, which is $\beta(B - \frac{L}{R})$, i.e., there is higher value for higher throughput. β is a constant of proportionality. Again, this is an approximation of reality, and not an exact measure.

– Without loss of generality, we set $\alpha = 1$, $\beta = 1$, and $B = 200$ (requests/second). While in a practical situation, α and β would be determined by policy, the shape of the overall optimization problem is exactly the same as if they were just set to 1. Based upon this, we obtain a total net value $N = V - C = B - \frac{L}{R} - R$, where N represents some monetary value.

The model is based on a scenario of *diminishing returns*, in which as resources are added, there is a point where adding resources increases cost more than value. In our scenario, in increasing the resource usage without the increase in other parameters, the total net value produced will be lower. As $V = B - \frac{L}{R}$ gets closer to B, resource utilization does not justify value. That means that total net value $N = V - C = B - L/R - R$ has a local minimum that is also the global minimum. Different hardware architectures determine different baseline performance values B, which do not affect the method for assuring optimal system performance. To maximize $N = V - C$, we estimate the derivative dN/dR and try to achieve the resource level which corresponds to $dN/dR = 0$, through a simple hill-climbing strategy. If $dN/dR > 0$, we increase R; if $dN/dR < 0$, we decrease R.

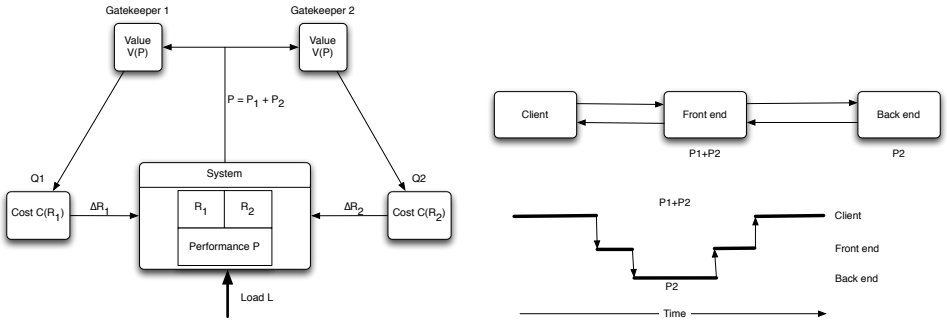
We chose this simple model as an approximation of reality that allows us to compare our algorithms with optima that we can compute. In reality, in our scheme, the optimum values for R are not known at runtime. It is often the case that in a practical situation, the reward curve follows a pattern of diminishing returns. For example, in adding resources, the user cannot necessarily perceive the difference. In our model, this is quantified by balancing cost and value, so that diminishing returns become apparent.

This differs from the standard model of assuring fixed setpoints for performance (as defined in SLOs or SLAs), in that there is a balance between cost and value rather than a specific goal. In our model, the setpoints are determined dynamically; if cost and value change, and the setpoint is invalidated, and our model instantly adjusts to a new global optimum, which has the character of a new setpoint.

3.2 Two closure operators

In earlier studies ([3], [4], [5]) we extended the closure model above to apply to the scenario of two closure operators, each controlling a separate resource variable influencing the same system. In this model the system delivers a service with a total response time $P = P_1 + P_2$, where P_1 (P_2) is the individual response time of the part of the system controlled by closure Q_1 (Q_2). In this case the overall value is $V = P = P_1 + P_2 = B - \frac{L}{R_1} - \frac{L}{R_2}$. Both closures receive the same feedback, which means that they are less able to identify the effects of their own actions.

The two-operator scenario is illustrated in Figure 1a and the corresponding closure model in Figure 1b. “Gatekeeper” nodes are responsible for gathering statistics on load and value, while “closure” nodes Q_1 and Q_2 control state based upon what the gatekeepers measure. This sets values for resources R_1 and R_2 in the system being managed. A typical web service contains “front end” and “back end” components that frequently compete for resources.



(a) Front-end/back-end system (b) The closure model

Fig. 1: The two-closure model

4 Information Exchange

The main motivation in this study was to determine which type of information that has the strongest effect on the precision of the closure model. The crucial part in the simulator is the estimation of dV/dR, which is the basis for estimating dN/dR. dV/dR is estimated through a linear curve fitting process, using available information observed in the environment. It is natural to assume that the more information used in this process, the better estimate we will obtain. In this study we compare different estimators of "slopes" based on selected variables that are observed over time. In this section the different slope-estimates will be explained, both for the single- and two-operator model.

4.1 Information exchange in the univariate case

The interpolation functions in the simulator are all based on a linear fit of the history of values of V and R to give an estimate of dV/dR. Since dC/dR is 1, dN/dR is $dV/dR - 1$. For the univariate scenario we have tested two different fitting functions:

1. *Minimum information:* We assume knowledge of the inverse nature of the relationship between V and R, such that we use linear interpolation to estimate a in $V = a\frac{1}{R} + b$.
2. *Full information:* Additional information of the system load L is required to make a linear fit of $V = a\frac{L}{R} + b$, which includes everything observable in our model.

4.2 Information exchange - two closure operators

We performed the same study for the model with two closure operators. In the model we tested, the system consists of two separate parts, each of which has an individual response time P_1 (P_2), but each of the closures receives the same

value feedback based on the overall response time $P = P_1 + P_2$. This makes it challenging to estimate their individual influence on the system based on changes in their own resource variable. In the multivariate scenario, three different slope estimators were tested.

1. *Independent optimization:* fits V to $a\frac{1}{R_i} + b$ for each of the closures i. This requires information about V and R_i for each closure. (I.e. does not require information about the other closure.)
2. *Knowledge of other resource use:* fits V to $a\frac{1}{R_1} + b\frac{1}{R_2} + c$. This requires information of V and both resource values R_1 and R_2.
3. *Full knowledge of resources and loads:* fits V to $a\frac{L}{R_1} + b\frac{L}{R_2} + c$. This requires information about V, R_1, R_2 and L.

5 Experiments and Results

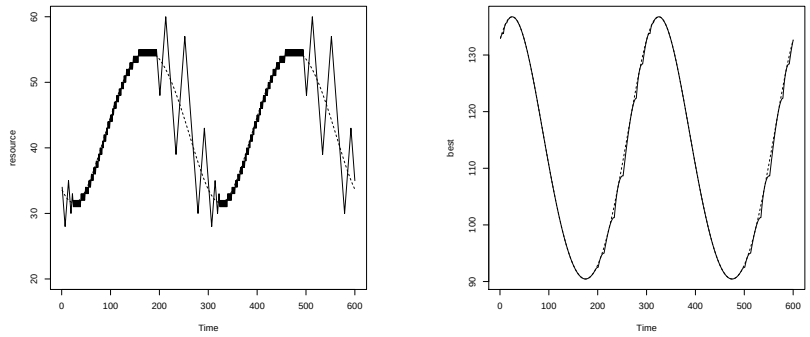
In this section the experiment setup will be explained, along with the main findings. We ran simulations on both the single-operator model and the two-operator model. For each of the model we ran simulations with different information exchange, two different levels for the single operator scenario, and three different dV/dR estimation methods for the two-operator scenario.

The decision-making process depends on received feedback on the perceived value V of the current system performance. The change in V is estimated through a sliding window computation. The "measurement window" is the number of measurements utilized in each prediction. This window has a finite size in measurements, where each measurement is done at a different time step. At each measurement step, the earliest measurement is discarded and replaced by a current measurement. Larger windows incorporate more history, which makes predictions more accurate in stable situations and less accurate in changing conditions. Smaller windows are more reactive, and adapt better to changes in situation. To check how the amount of available history affected the precision of the models, we varied the size of the measurement window $w = 3, 5, 10$.

System load was sinusoidal, $L(t) = 1000sin((t/p) * 2\pi) + 2000$, which made the load vary periodically between 1000 and 3000. Many realistic sites observe roughly sinusoidal load variation based upon day/night cycles. We recorded resource usage, response time and net value for all simulations.

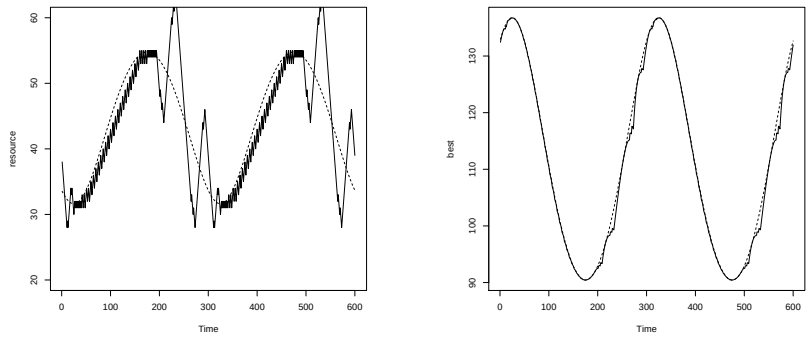
When the closures do not exchange full information, i.e. when the closures do not use information about system load and the resource level of the other operators, we observe what we refer to as a *hidden variable problem.* The results from both the single- and two-closure operator model show that the precision of fit of the model compared to the theoretical values have been lower in certain parts of the data. In simulation results (like Figure 2), when the load L increases, the closure model produces estimates of R that are quite close to optimal. However, when load decreases, the optimality of the solution is lower. The R estimates oscillate around the optimum, but move quite far away from the optimum curve.

Increasing the measurement window did not have any positive effect, as seen in Figure 3 and Figure 5. Up-hill, the resource usage curve is shifted to the



(a) Resource usage (b) Net value

Fig. 2: Single operator model. Resource usage and net value for the minimum information scenario. $w = 3$.

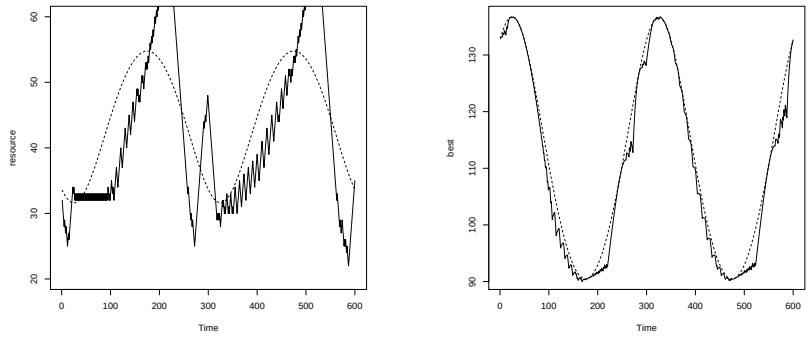


(a) Resource usage (b) Net value

Fig. 3: Single operator model. Resource usage and net value for the minimum information scenario. $w = 5$.

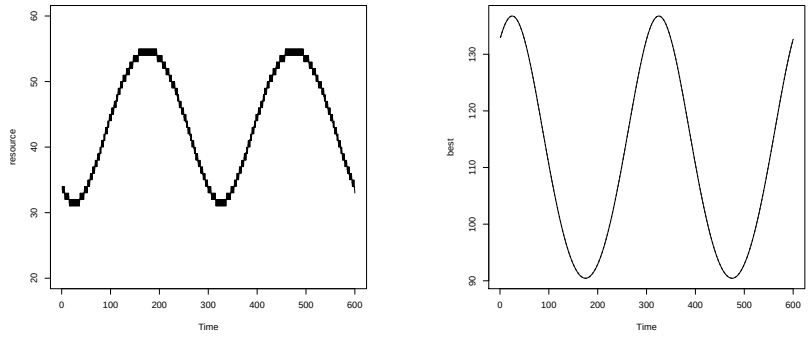
right of the optimum curve, while the oscillations increase downhill compared to when the input window is smaller. This is evidence that the actual resource usage varies farther from the optimum curve for larger input windows.

To mitigate the oscillation problem, we must add "full information" (Figure 5). For the single operator model, this includes information about the inverse relationship between V and R, and the value of system load L. In this case the net value tracks the optimum value quite closely. Clearly, this performs better than the partial information model.



(a) Resource usage (b) Net value

Fig. 4: Single operator model. Resource usage and net value for the minimum information scenario. $w = 10$.

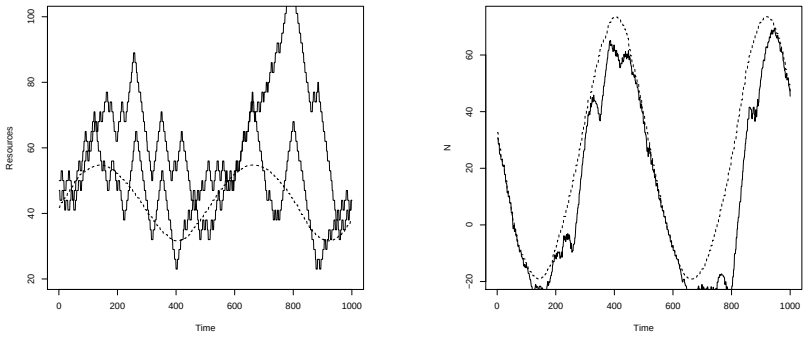


(a) Resource usage (b) Net value

Fig. 5: Single operator, full information, $w = 3$. Adding information about L (b)) removes the oscillation as load decreases.

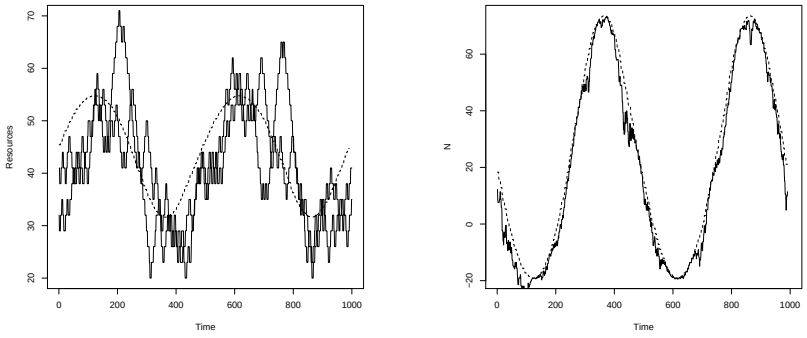
For the two-closure model, full information includes all information for the single operator model, plus information about R_1 and R_2; which means that the agents exchange information about their current resource levels.

The simulations show that heavy oscillation in resource usage is present when we do not provide full information. The estimator that results in the worst performance, is independent optimization (Figure 6), in which the two operators optimize separately without exchanging any information. The model performs better when the resource demand increases, but resource allocations oscillate extensively when the resource demand decreases (Figure 6a). As seen in Figure



(a) Resource usage (b) Net value

Fig. 6: Results using independent optimization.



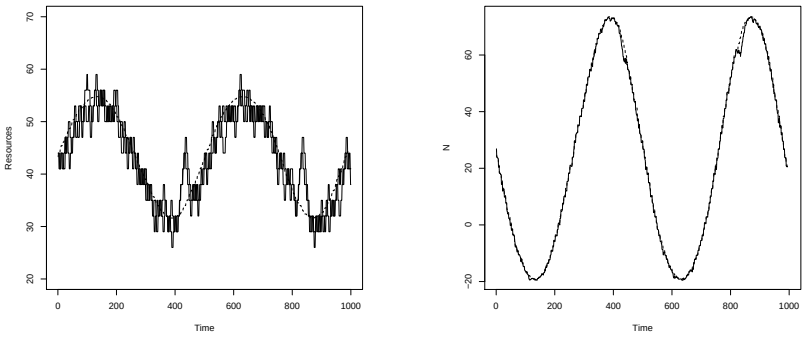
(a) Resource usage (b) Net value

Fig. 7: Results using knowledge of other resource use.

6b, this affects net value, which varies far from the theoretical best (the dotted line).

Adding information about the second operator improves performance somewhat (Figure 7). There is still more oscillation when load decreases (Figure 7a), but significantly less compared to the results for independent optimization. The improvement is more obvious when comparing net value N in the two cases (Figure 7b).

Adding additional information about system load more or less removes the oscillation effect (Figure 8). The resource usage generated by the simulator tracks the theoretical optimum with high precision (Figure 8a), and the curve representing total net value is almost precisely the theoretical optimum.



(a) Resource usage (b) Net value

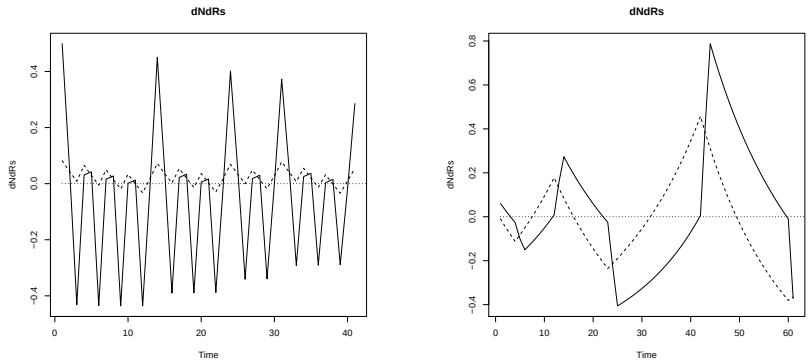
Fig. 8: Results using full knowledge of resources and loads.

To obtain an understanding of what generates the oscillations in the experiments, we studied how the model estimates dN/dR. For the scenario illustrated in Figure 2, we plotted the values for dN/dR estimated by the simulator. Figure 9a shows the estimated dN/dR-values (solid line) for a part of the simulation when the load L increases, while Figure 9b displays the same values when load decreases. The dashed line shown in both figures represents the theoretical value of dN/dR, which is $\frac{L}{R^2} - 1$. The horizontal line in each figure determines whether resources will be incremented (for dN/dR above the line) or decremented (for dN/dR below the line).

Figure 9b shows a delay in tracking the correct dN/dR-values, which is caused by the sliding window. As data on the change enters the window, it takes a few steps of measurement for the prediction to change. This creates greater and greater differences from optimal that the fixed resource increment size never corrects, even when resources are being updated in the proper direction. The delay causes a deviation from the theoretical values, and this deviation increases as the simulation progresses. This suggests that that implementing adjustable increments based upon the relative magnitude of the estimated dN/dR could solve the oscillation problem.

6 Conclusions

We discovered a phenomenon of heavy oscillation in our closure model. For the model to estimate the optimal resource level throughout the simulations with minimum error, information about system load is crucial. Removing that information from the model does not significantly affect the case when load is increasing, but when load decreases, not accounting for its decrease causes oscillation away from the optimum. Thus the load is a "hidden variable" that – when exposed – improves adaptive behavior.



(a) dN/dR-values when load increases.

(b) dN/dR-values when load decreases.

Fig. 9: The estimation of dN/dR-values in the single operator scenario, minimum information.

The oscillations around optimum disappear when full information is used in the decision-making process. Full information is defined as current resource values in both controllers and system load. In the single-operator scenario, even minimum information gives total net value quite close to the theoretical maximum.

For the two-operator scenario, the downhill-oscillation effect is significantly worse for the *independent optimization*-method, which is the interpolation method that does not use information about both operators. The hidden variable effect is stronger when each agent makes decisions without taking other agents into account. The oscillation disappears when we add load information into the decision mechanism.

Finally, the oscillations seem to be caused by a combination of the fixed window size and the fact that resources are always changed by a fixed amount. Detailed analysis of dN/dR values suggests that a varying resource increment size based upon the relative magnitudes of dN/dR may solve the oscillation problem without adding additional information.

References

1. Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. Kingfisher: Cost-aware elasticity in the cloud. In *INFOCOM, 2011 Proceedings IEEE*, pages 206–210. IEEE, 2011.
2. Pablo Chacin and Leando Navarro. Utility driven elastic services. In *Distributed Applications and Interoperable Systems*, pages 122–135. Springer, 2011.
3. Siri Fagernes and Alva L Couch. On the combined behavior of autonomous resource management agents. In *Mechanisms for Autonomous Management of Networks and Services*, pages 38–49. Springer, 2010.

4. Siri Fagernes and Alva L Couch. On alternation and information sharing among cooperating autonomous management agents. In *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on*, pages 364–369. IEEE, 2010.
5. Siri Fagernes and Alva L Couch. Coordination and information exchange among resource management agents. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 422–429. IEEE, 2011.
6. Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259, 2006.
7. Daniel F Macedo, Aldri L dos Santos, José Marcos S Nogueira, and Guy Pujolle. A knowledge plane for autonomic context-aware wireless mobile ad hoc networks. In *Management of Converged Multimedia Networks and Services*, pages 1–13. Springer, 2008.
8. Maïssa Mbaye and Francine Krief. A collaborative knowledge plane for autonomic networks. In *Autonomic communication*, pages 69–92. Springer, 2009.
9. David D Clark, Craig Partridge, J Christopher Ramming, and John T Wroclawski. A knowledge plane for the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–10. ACM, 2003.
10. Pradeep Padala, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, Arif Merchant, and Kenneth Salem. Adaptive control of virtualized resources in utility computing environments. *ACM SIGOPS Operating Systems Review*, 41(3):289–302, 2007.
11. Giovanni Pacifici, Wolfgang Segmuller, Mike Spreitzer, and Asser Tantawi. Dynamic estimation of cpu demand of web traffic. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 26. ACM, 2006.
12. Constantin Adam and Rolf Stadler. Service middleware for self-managing large-scale systems. *Network and Service Management, IEEE Transactions on*, 4(3):50–64, 2007.
13. Ahmed Ali-Eldin, Johan Tordsson, and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.
14. Alva L Couch and Marc Chiarini. Dynamics of resource closure operators. In *Scalability of Networks and Services*, pages 28–41. Springer, 2009.
15. Alva L Couch and Marc Chiarini. Combining learned and highly-reactive management. In *Modelling Autonomic Communications Environments*, pages 1–14. Springer, 2009.