Pricing the Cloud: An Adaptive Brokerage for Cloud Computing

Philip Clamp and John Cartlidge
Department of Computer Science
University of Bristol
Bristol, UK
Email: phil@clamped.me.uk, john@john-cartlidge.co.uk

***Abstract*—Using a multi-agent social simulation model to predict the behavior of cloud computing markets, Rogers & Cliff (R&C) demonstrated the existence of a *profitable* cloud brokerage capable of benefitting cloud providers and cloud consumers alike. Functionally similar to financial market brokers, the cloud broker matches provider supply with consumer demand. This is achieved through *options*, a type of derivatives contract that enables consumers to purchase the *option*, but not the *obligation*, of later purchasing the underlying asset—a cloud computing virtual machine instance—for an agreed fixed price. This model benefits all parties: experiencing more predictable demand, cloud providers can better optimize their workflow to minimize costs; cloud users access cheaper rates offered by brokers; and cloud brokers generate profit from charging fees. Here, we replicate and extend the simulation model of R&C using *CReST*—an open-source, discrete event, cloud data center simulation modeling platform developed at the University of Bristol. Sensitivity analysis reveals fragility in R&C's model. We address this by introducing a novel method of autonomous adaptive thresholding (AAT) that enables brokers to adapt to market conditions without requiring *a priori* domain knowledge. Simulation results demonstrate AAT's robustness, outperforming the fixed brokerage model of R&C under a variety of market conditions. We believe this could have practical significance in the real-world market for cloud computing.**

***Keywords*—CReST; simulation; cloud computing; brokerage**

I. Introduction

Cloud computing is the latest step change in the delivery of computing services as a utility—a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources. Migration to the Cloud involves users moving the location of local compute infrastructure to the network, thereby reducing costs commonly associated with managing hardware and software assets, and gaining from the economies of scale enjoyed by cloud providers [1], [2].

The term "cloud computing" encapsulates both the applications delivered *"as a Service"* and the underlying hardware and software infrastructure in the ultra-large scale data centers that make the concept viable [3]. This infrastructure is commonly known as a *Cloud* and can be *public*, *private*, or a *hybrid* of the two, while a service application delivered to end users is often referred to as *Software as a Service* (SaaS), *Platform as a Service* (PaaS), or *Infrastructure as a Service* (IaaS), depending on which level of the software stack is provided. SaaS typically describes end user applications that are accessed remotely over the internet and includes ubiquitous software applications such as GoogleMail, Facebook and Twitter. IaaS describe lower-level applications that offer users access to the underlying cloud hardware via a virtualization layer. Typically, for IaaS, users purchase Virtual Machine (VM) instances that are configured with an operating system and offer access to virtual CPU, RAM and hard disk storage. These VMs can then be configured by the user to provide the specific functionality required. From the user's perspective, VM instances are exactly the same as their own physical hardware accessed remotely. Finally, at the intermediate level between SaaS and IaaS, PaaS offers a suite of software tools and interfaces—a *platform*—upon which users can build and integrate their own software applications. Currently, the clear trend of providers offering ever more bespoke infrastructure products, means that the distinction between PaaS and IaaS is no longer clear (for example, AWS's RDS Database instance). However, for clarity, in this paper, when we consider cloud resources, we refer to IaaS VM instances and *not* the higher-level software applications (Facebook, Twitter, etc.) that are built on top.

The *on-demand* delivery model for cloud computing resources offers a variety of benefits for business consumers [3]. The ability to start and stop VM instances almost instantly, when required, gives enormous flexibility and scale-out opportunities. In addition, businesses no longer need to invest capital resources in purchasing the often underutilized compute infrastructure needed to cover peak business demand; including all additional costs such as support staff and maintenance [3]. However, the on-demand pricing model is not necessarily ideal for cloud providers, as they attempt to adhere to strict Service Level Agreements (SLA) in the face of fluctuating demand. If providers could accurately forecast, or have advanced knowledge of, future resource demand, then they would have the opportunity to reduce costs by optimizing electricity purchases, engineering staff, and hardware utilization, etc.

At present, most providers offer a *fixed price* model where Virtual Machine (VM) instances are purchased for a fixed time period (*reserved instances*), or billed per hour of usage (*on demand*). Some providers, e.g., Amazon Web Services (AWS), offer an alternative *spot price* tariff that varies in real-time based on current supply and demand [4]. However, of these methods, only long-term reserved instances (maximum 36 months) aid the provider in capacity planning. Several alternative pricing models have been proposed in academic research, most notably involving *derivatives contracts*, such as (European) *options* [5]. Options contracts involve the payment of an up-front fee that gives the buyer the legal right, but not the obligation, to purchase a resource for an agreed strike-price on some later delivery date [6]. These types of financial instruments are commonly used in financial commodities markets where their underlying assets range from wheat and oil, to a suite of complex financial products.

In their investigation into cloud computing pricing models, Rogers & Cliff (R&C) used an agent-based simulation model to explore the possibility of a cloud computing services

broker delivering derivative contracts to provide both cheaper resources to consumers *and* aid providers in predicting future usage [4]. They invariably found that not only was it possible to do this, but that in addition the broker was able to generate a significant profit. R&C's result has the potential to significantly impact the delivery and pricing of cloud services. As the market in cloud resources matures and becomes more standardized, the promise of a *federated cloud*—where cloud users can migrate between providers seamlessly—will theoretically allow resources to be traded as a commodity; eradicating existing concerns of vendor lock-in. In turn, this will open opportunities for brokers to enter the market, acting as intermediary *market makers* between users and providers. In such a scenario, R&C's result could have practical as well as academic significance. In this paper, we attempt to replicate and extend the work of R&C. We show that R&C's results are sensitive to model parameter settings and require *a priori* information to maximize profitability. By introducing a novel adaptive learning process, we offer a robust solution to this problem, enabling the broker to automatically maximize profit under a range of market conditions.

This paper is organized as follows. In Section II, we introduce the cloud brokerage model [4] used by R&C to demonstrate the possibility of a profitable broker acting as a third-party mediator between cloud users and cloud providers. In Section III, we briefly introduce CReST—a cloud simulation platform that we use for our empirical simulations—and detail our experimental assumptions and configuration. We then perform three sets of experiments. Firstly, in Section IV we replicate the work of R&C [4] to verify the validity of our simulation model design. Then, to test the robustness of the conclusions drawn by R&C, in our second set of experiments (Section V) we perform a sensitivity analysis on R&C's model. Subsequently, having demonstrated the sensitivity of R&C's optimal threshold value, θ_{opt}, we extend the brokerage model of R&C by introducing a novel method for automatically adapting θ (AAT) during run-time. Our third and final set of experiments (Section VI) demonstrate the performance of AAT under a variety of market conditions. We show that AAT is able to automatically find θ_{opt} under a variety of market conditions with no *a priori* information. Finally, in Section VII we conclude that AAT is a significant, robust extension to R&C's model and one that may have practical significance in the real-world market for cloud computing resources.

II. Background: R&C's Brokerage Model

Typically, the role of a broker is to facilitate the matching of supply and demand in a market. Brokerage services primarily generate profit by charging commission fees, and/or *making the spread* by buying at a lower price and selling at a higher price. In the cloud brokerage model of Rogers and Cliff (R&C) [4], the broker aims to make a profit by purchasing long-term advanced obligations on resources (36 month reserved instances), and repackaging them as 1 month options contracts that they sell at a higher price to users.

The brokerage model of R&C consists of two stages: (1) each month, the broker takes orders from clients for future resource needs by selling options, and determines how many reserved instances to purchase; (2) in the following month, clients can request instances from the broker by *exercising* their options. If the broker has capacity available from previously purchased reserved instances, they can sell it on to users at a profit. Otherwise, the broker must purchase additional (more expensive) on-demand instances from the provider to fulfill the obligation of the client.

R&C's brokerage model follows a pricing structure that was initially developed at HP Labs by Wu, Zhang, and Huberman (WZH) [5]. The WZH model financially rewards clients that reveal the *true likelihood* that they will utilize a resource in the future. Each month, every client, i, estimates their own probability, p_i, of using a resource in the following month. Clients then submit their estimation, p_i, to the broker in order to purchase a resource option. In the following month, the client is charged $Used(p_i)$ if the option is exercised (i.e., if the resource is used) and $Unused(p_i)$ if the option is not exercised (i.e., if the resource is not used), such that:

$$Used(p_i) = 1 + \frac{k}{2} - kp_i + \frac{kp_i^2}{2} \tag{1}$$

and

$$Unused(p_i) = \frac{kp_i^2}{2} \tag{2}$$

where $k = 1.5$ in [5]. If users choose instead to purchase resources directly from the provider, they will expect to pay Op_i, where O is the on-demand cost of a one-month instance (in the original model, $O = 2$ [5]). We can consider this contract as an options model if the broker charges clients $Unused(p_i)$ to purchase the option contract and then a further charge of $Used(p_i) - Unused(p_i)$ in the following month if the option is exercised (if the resource is used). The model can be calibrated to real-world prices by multiplying $Used(p_i)$ and $Unused(p_i)$ by a *cost factor* [4]. It has been proven that this pricing model encourages users to truthfully submit their honest estimate of resource usage, p_i [5].

Each month, once the broker has sold options contracts (and has thus received probability, p_i, estimates from clients), the broker must decide whether or not to purchase additional long-term (36 month) reserved instances from the provider. If the broker has previously purchased enough reserved instances to cover the predicted demand, Σp_i, no further instances are purchased. However, if the broker does not own enough reserved instances to cover expected demand, additional reserved instances are purchased using the following algorithm [4]. Firstly, the broker observes historical resource demand, $\boldsymbol{H} = [h_{t-36}, \ldots, h_t]$, over the previous 36 month period, and compares against the future resource capacity, $\boldsymbol{F} = [f_t, \ldots, f_{t+36}]$, (the number of reserved instances owned) over the forthcoming 36 month period. Using a simple forecasting mechanism that assumes future demand will equal previous demand lagged 36 months, the broker then calculates an *expected deficit profile*, $\boldsymbol{D}$, for each forthcoming month by subtracting historical demand, $\boldsymbol{H}$, from future capacity, $\boldsymbol{F}$, for each month, such that:

$$\boldsymbol{D} = \boldsymbol{F} \text{ - } \boldsymbol{H}. \tag{3}$$

For each resource required, the *Marginal Resource Utilization (MRU)* is the proportion of months in $\boldsymbol{D} > 0$. The *MRU* estimates the fraction of life ($months/36$) an additional reserved instance is likely to be utilized over the next three

years, based on historical demand. Brokers then use a threshold, θ, to determine whether or not to purchase a new 36-month reserved instance. If $MRU > \theta$, the broker buys a new instance, estimating that it will be used in enough months to make a profit. Alternatively, if $MRU < \theta$, the broker does not purchase a new instance, estimating that it will be underutilized and that purchasing on-demand monthly instances, when necessary, will be more profitable. Each month, the broker delivers 1-month access to reserved instances to clients that exercise their options. If the broker does not have the capacity to fulfill client demand, they purchase additional on-demand instances directly from the provider. In general, the monthly purchase cost of on-demand instances is greater than the monthly cost of 36-month reserved instances. R&C demonstrated that this model can generate broker profits while also benefitting users and providers [4]: users access cheaper monthly resource costs and providers sell a greater proportion of 36-month reservations, aiding in capacity planning to reduce provision costs. For more detailed description of R&C's brokerage model, we refer the reader to [4].

III. Simulation Methodology

The Cloud Research Simulation Toolkit (CReST) was developed at the University of Bristol to address the need for a robust simulation modeling tool for research and teaching of data center management and cloud provision. CReST is a stand-alone application, written in Java, and is freely available open source under a GNU General Public License v3.0 [7]. Although alternative tools exist, CReST has a unique feature set (see [8]) that enables simulation at multiple abstraction levels: from physical hardware, energy usage and thermal flows within a DC, to networked infrastructure and the virtualization layer of application services supporting dynamic user demand. For details on the architecture of CReST, refer to [8].

For all experiments reported in this paper, we use CReST as the cloud simulation platform. CReST is designed as a set of coupled *modules* that can be independently switched on or off depending on the level of abstraction required. Here, to optimize simulation performance, we disabled several of the lower-level physical infrastructure modules, such as the *Thermal* module that tracks air-flow in the data center. The active modules used in all of the brokerage simulations that we perform include: *Brokerage*, *Pricing*, *Events*, *Services* and *Simulation*. This enabled us to efficiently run experiments that simulate decades of time, without compromising on the abstraction level needed. All CReST code used to run the experiments performed here, and associated Python scripts used for data analysis and visualization, are available to download in version 0.3.0 of CReST [7].

The parameter space used for all experiments, unless otherwise stated, are detailed below:

- **Running Time:** Each simulation lasts 276 simulated months. This time period is determined by the available demand data utilized by R&C (refer to Fig. 1).
- **Number of User Agents:** Following R&C, we set the number of agents that demand resources to 1000.
- **Pricing:** Prices for cloud computing instances in the real world undergo continual change due to underlying factors such as hardware costs and competition. For the R&C replication experiments (Section IV), we follow the same pricing scheme as [4]. In later experiments (Sections V and VI), we use real-world prices charged by Amazon Web Services (AWS).
- **Reservation and Learning Period Length:** R&C explored 12 and 36 month reservations and demonstrated similar results, but increased broker profits for 36 months [4]. Here, we use only 36 month reservations.
- **Cost Factor:** The WZH charging model [5] is based on reservations with a cost of 1 or 2 and therefore needs to be scaled in order to simulate AWS pricing. In R&C's previous work, the cost factor, C, has varied (i.e., 35 [4] and 60 [9]). In Section IV, we use a cost factor of $C = 35$ to replicate R&C. Then, in Section V, we explore the sensitivity of R&C's model by varying this cost factor.
- **Demand Profiles:** Following [4], to simulate *realistic* demand for virtual machines, we consider four demand profiles generated using real demand data over the period 1988-2011 for a variety of IT-related industries. This data set was collated by Owen Rogers, using the UK Office for National Statistics' database of Non-Seasonally Adjusted Index of Sales. Fig. 1 displays the four demand profiles that we label using R&C's terminology: *Rapid Growth* (top-left), *Steady Growth* (top-right), *Recession & Recovery* (bottom-left) and *Steady* (bottom-right). These data were supplied to us by Owen Rogers to enable us to perform a strict replication of R&C's experiments [4], [9]. For further details on the collection and rationale of data, refer to [4].
- **Marginal Resource Utilization Thresholds:** In Sections IV and V, we explore a range of thresholds, θ, to determine the optimal (most profitable) value, θ_{opt}, under a variety of market conditions. In Section VI, as an extension to R&C's model, we introduce an autonomous adaptive thresholding technique (AAT) that automates the selection of θ during runtime.

Each experiment was repeated 30 times to enable statistical hypothesis testing of the results. All code used for experiments detailed in this paper is available to download in CReSTv0.3.0 at https://sourceforge.net/projects/cloudresearch/.

IV. Replication of R&C's Brokerage Model

In [4], R&C use an exhaustive search to determine the optimal Marginal Resource Utilization (MRU) thresholds, θ_{opt}, for each of the four markets shown in Fig. 1. They show that θ_{opt} varies between markets and that, when using θ_{opt}, the broker maximizes profit. They further show that all values of $\theta < 1.0$ generates a profit for the broker in all markets, even when $\theta = 0$; i.e., in the trivial case where the broker will *always* purchase an additional reserved 36-month instance whenever there is a new unit of expected demand. When $\theta = 1.0$, the broker will *never* purchase a reserved instance, hence profits are always 0.

In this section, we replicate the model of R&C as closely as possible in order to: (1) determine whether R&C's results

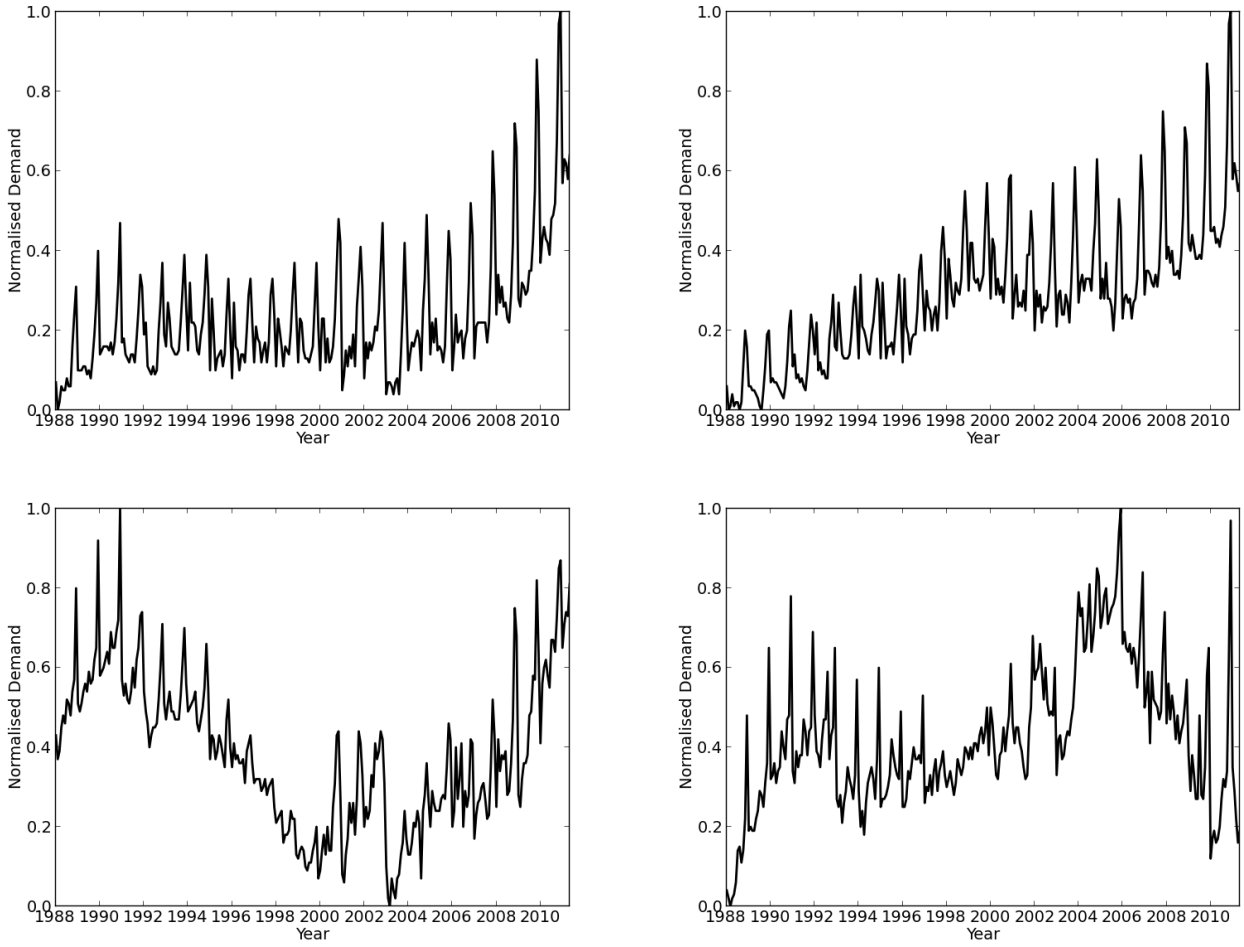

Fig. 1. Normalized demand profiles for the period 1988-2011, labeled: *Rapid Growth* (top-left); *Steady Growth* (top-right); *Recession & Recovery* (bottom-left); and *Steady* (bottom-right). For details, refer to [4].

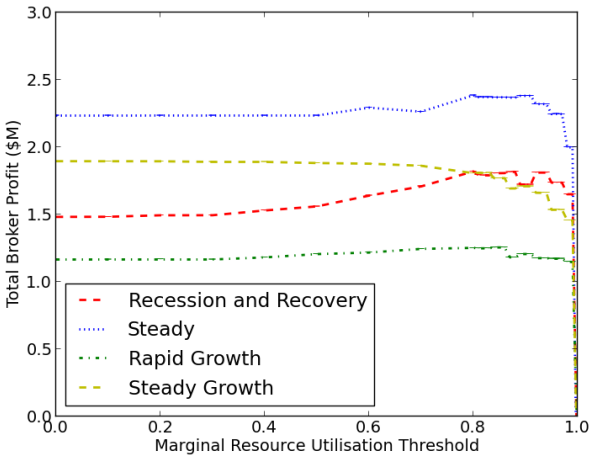

Fig. 2. Total broker profit in $Millions (mean $\pm 95\%$ CI, 30 runs) for each market across different thresholds, θ, using 36 month reserved instances. The resolution of θ between 0.0-0.8 is 0.1 and between 0.8-1.0 is 0.01.

are repeatable; and (2) verify and validate our CReST implementation of R&C's model. To perform this replication, we exhaustively tested a subset of MRU thresholds, $0.0 \leq \theta \leq 1.0$, to determine the profitability of each strategy in each of the four markets shown in Fig. 1. Results are plotted in Fig. 2. To reduce the search space, eleven θ thresholds were initially tested, such that $\theta \in \{0.0, 0.1, \ldots, 1.0\}$. Performing these simulations using 4 market profiles and repeating each trial 30 times meant a total of $11 \times 4 \times 30 = 1320$ simulation runs. Then, having noticed that the turning point for many of the profit curves in Fig. 2 were in the region of 0.9, an additional set of runs were performed at a resolution of 0.01, such that $\theta \in \{0.81, 0.82, \ldots 0.89, 0.91, 0.92, \ldots 0.99\}$. This led to an additional $18 \times 4 \times 30 = 2160$ simulation runs.

In Fig. 2, we see broker profits (mean of 30 runs $\pm 95\%$ confidence interval displayed using vertical bars) for each market, plotted as a function of θ. For *Steady* (blue dots), *Recession & Recovery* (red dash), and *Rapid Growth* (green dot-dash) markets we see broker profits increase with θ until a turning point in the region $\theta \approx 0.9$. However, in the *Steady Growth* market (yellow dash), profits gradually fall as θ rises, until $\theta \approx 0.8$, after which profits rapidly decline. For all markets, when $\theta = 1.0$ brokers make no profit (as expected).

TABLE I. COMPARISON OF BROKER PROFITS ($MILLIONS) ACROSS MARKETS. R&C'S ORIGINAL RESULTS [4] ARE PARENTHESIZED.

Market	θ_{opt}	36 Month Reservations Profit ($M)		
		$\theta = 0$	$\theta = \theta_{opt}$	$(\theta_{opt} - \theta_0)\%$
Rapid Growth	0.84 (0.72)	1.17 (1.15)	1.26 (1.27)	7.7% (10.4)
Steady Growth	0.00 (0.00)	1.89 (1.85)	1.89 (1.85)	N/A (N/A)
Recession & Recovery	0.80 (0.80)	1.48 (1.48)	1.82 (1.80)	23.0% (21.6)
Steady	0.91 (0.82)	2.23 (2.22)	2.38 (2.45)	7.1% (10.4)

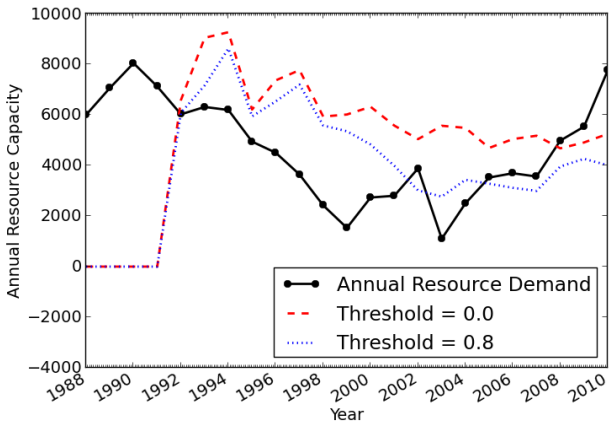

Fig. 3. Annualized broker-owned resources versus demand in a Recession & Recovery market. When $\theta = 0.8$ (blue-dot), the broker's resource purchases more closely track demand (black-line) than when $\theta = 0.0$ (red-dash).

Further, for all markets, brokers make a profit for *all* values in the range: $0.0 \leq \theta < 1.0$. These results are *qualitatively* similar to those published by R&C [4].

Table I presents a detailed quantitative comparison of results against the original results of R&C [4]. For each market we tabulate: (1) the optimum threshold value (θ_{opt}); (2) the mean profit for brokers that *always* purchase an additional reserved instance ($\theta = 0$); (3) the mean profit for brokers that use the optimum MRU threshold ($\theta = \theta_{opt}$); and (4) the percentage difference in profit between brokers using the optimum threshold value and brokers that always purchase a new instance, i.e., the percentage difference between the previous two columns ($\theta_{opt} - \theta_0$). Values in parentheses are the values obtained by R&C [4]. We see that there is a strong quantitative similarity. All profits are within 5% of the values presented by R&C (indeed, most are within 2.5%). Further, for the optimal threshold values, θ_{opt}, two are identical (*Steady Growth* and *Recession & Recovery*), one is within 10% (*Steady*) and one is within 20% (*Rapid Growth*). As shown in Fig. 2, the *profit gradient* is very shallow in *Rapid Growth* markets (green dot-dash), meaning that profit is relatively insensitive to θ, hence this is the market that we would expect the most discrepancy in results. Overall, we believe that these results demonstrate a strong *quantitative* replication of R&C.

In Fig. 3, we plot the annual broker-owned resource capacity against market demand for two example simulation runs in a *Recession & Recovery* market with MRU thresholds $\theta = 0.0$ (red dash) and $\theta = \theta_{opt} = 0.8$ (blue dots). We see that the optimal θ value (blue dots) more closely tracks actual resource demand (black line), resulting in a greater utilization of purchased 36-month reserved instances. When the broker *always* buys additional instances (red dash), brokers end up purchasing too much capacity, which goes largely underutilized—the area bounded by the red (dash) and black lines from above and below, respectively. This figure demonstrates how tuning the value of θ can enable broker capacity to more closely match user demand, thus maximizing utilization and ultimately maximizing profits. In the majority of markets (*Steady Growth* is the obvious exception), the optimal thresholds, θ_{opt}, tend to be relatively high, falling in the region > 0.8 (and in R&C's original results, in the region > 0.72). This suggests that it is more risky for the broker to purchase a significant number of reserved instances that go underutilized, than it is to purchase fewer and risk buying more expensive on-demand instances. This is not true in the *Steady Growth* market ($\theta_{opt} = 0.0$), where it is *always* beneficial to buy an additional instance since continual market growth guarantees resource utilization.

In this section, we have demonstrated that the cloud brokerage results of R&C are repeatable and verified that our replication of R&C's model using the CReST simulation platform is valid. In the following section, we perform a sensitivity analysis on the model to test the robustness of R&C's results.

V. SENSITIVITY ANALYSIS OF R&C'S BROKERAGE MODEL

In this section, we perform a sensitivity analysis of R&C's brokerage model to determine the robustness of results. In the previous section, we observed that the optimal MRU threshold, θ_{opt}, varies with market demand profile. Here, we analyze the sensitivity of θ_{opt} to other model parameters: (a) resource prices; (b) cost factor; and (c) demand variance.

A. Sensitivity to Provider's Resource Pricing

Here, we update the pricing of resources to reflect the current pricing tariff used by AWS (March 2013):

- Monthly on Demand = $46.80
- Up-Front Reserved = $250.00
- Monthly Reserved = $13.68

We repeated the experiments from Section IV using the pricing tariff, above. All other configuration parameters were unchanged, including the prices the broker charges clients (the cost factor). Results are presented in Table II. We see that across all markets the optimum threshold, θ_{opt}, is *lower*. Further, the additional profit gained by using the optimal threshold, θ_{opt}, rather than the zero threshold, $\theta = 0$, is much smaller, less than 1% in all markets (final column). This result demonstrates that θ_{opt} is sensitive to the provider's pricing tariffs. In the scenario simulated here, the broker has lower purchase costs (AWS's prices have fallen since R&C's original model). However, the broker does not pass these savings on to users. Hence, the broker's profit in each market increases (compare Table I with Table II). At the same time, the *risk* of purchasing a reserved instance that will be underutilized is lowered. Thus, across all markets θ_{opt} falls.

TABLE II. BROKER PROFITS USING CURRENT AWS PRICING.

Market	θ_{opt}	36 Month Reservations Profit ($M)		
		$\theta = 0$	$\theta = \theta_{opt}$	$(\theta_{opt} - \theta_0)\%$
Rapid Growth	0.4	1.50	1.51	0.67%
Steady Growth	0.1	1.93	1.93	0%
Recession & Recovery	0.6	2.19	2.21	0.91%
Steady	0.0	2.52	2.52	N/A

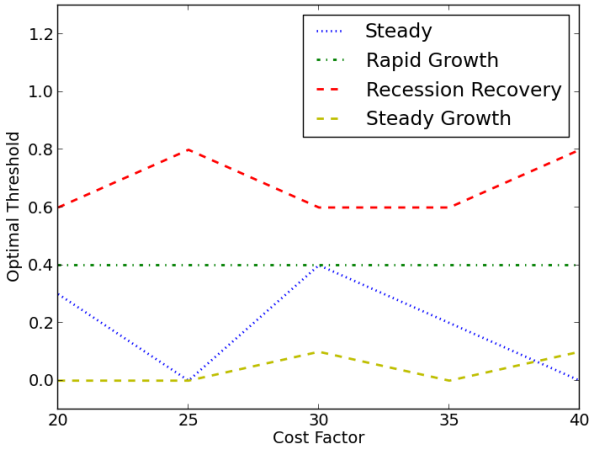

Fig. 4. Optimal thresholds, θ_{opt}, as a function of cost factor.

B. Sensitivity to Broker's Pricing

Here, we examine the effect of varying the prices that Broker's charge users. We control this by varying the *cost factor*, C (refer to Section II). As one would expect, the cost factor variable is directly related to broker profits, with higher cost factor producing higher profits.

Fig. 4 shows the response of θ_{opt} to changes in C. We see that in all markets, apart from *Rapid Growth*, θ_{opt} is sensitive to C and that this relationship is nonlinear.

C. Sensitivity to Variation in Demand

Here, we examine the effect of adding variance (noise) to the market demand profiles presented in Fig. 1. Results are presented in Fig. 5. We see that, in all markets, θ_{opt} is sensitive to variation in the demand profiles and that this relationship is nonlinear.

We have demonstrated that θ_{opt} is highly sensitive to the provider's pricing tariff, to the broker's pricing tariff, and to variation in demand. This confirms that the selection of an appropriate θ_{opt} value for the broker is a nontrivial task. Therefore, we propose that the value of θ should be dynamically adapted in real time in response to contemporaneous market dynamics. In the following section, we propose a novel method for such autonomous adaptive thresholding (AAT) and empirically test its utility. For all experiments, unless otherwise stated, we use the latest AWS pricing tariff presented in Section V-A. We also use a cost factor $C = 30$, selected to preserve the ratio between provider pricing and broker pricing as used in the original brokerage model of R&C.

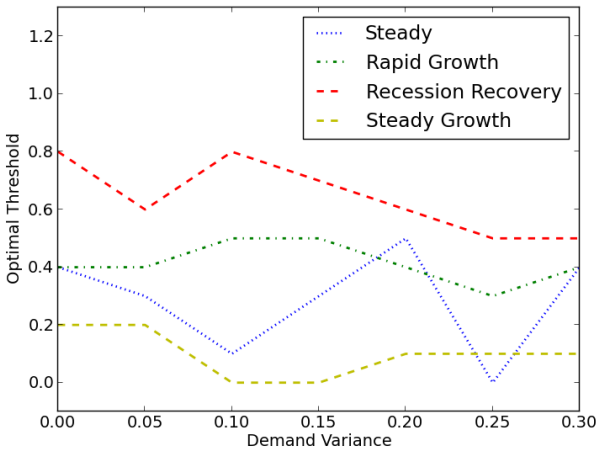

Fig. 5. Optimal thresholds, θ_{opt}, as a function demand variance.

VI. EXTENSION OF R&C'S BROKERAGE MODEL: AUTONOMOUS ADAPTIVE THRESHOLDING (AAT)

The evident sensitivity of the threshold parameter and its intrinsic contribution to the overall performance of the model presents a complication for the application in real world scenarios. Selecting the optimal θ value enables the broker to balance its asset exposure to the providers in a favorable manner, ultimately reducing risk and maximizing profits. The WZH Model leverages the data of past events in order to hedge risk appropriately. However, due to the nature of its operating environment it is not known *a priori* if the market will continue to follow the same pattern. Up to this point, the experiments conducted have been based on real world past data - however, the inherent unpredictability and vicissitudes of the world markets could render forecasts made on previous demand meaningless. A market shock where demand for a resource in the community suddenly alters, perhaps caused by a new entrant to a market, could lead to the broker operating with a suboptimal threshold parameter, leaving it risk exposed in the number of reservations currently owned. Doubtlessly, therefore it would be advantageous for θ to be automatically updated to reflect the current market circumstances during operation. Here, a versatile technique is presented that enables the broker to autonomously update θ online.

A. Formulation of AAT

The Autonomous Adaptive Thresholding mechanism (AAT) utilizes the Widrow-Hoff delta rule [10] to streamline the threshold selection between iterations (each month) of the hedging process. The delta rule is a general learning method that has been shown to be effective in a variety of domains such as Algorithmic Trading [11] and coevolutionary optimization [12]. The delta rule is one of the simplest rules in Machine Learning, forming the basis of both adaptation algorithms [13] and reinforcement in classifier systems [14], [15]. The delta rule attempts to minimize the error between a real system output and a target output determined by some domain-specific proxy. Using the projected reservation utilization as a proxy, AAT updates the θ value in each reservation stage of the model

through the minimization of the error between the current threshold and the determined target. If there is no error between the system output and the desired output, then no learning takes place. Conversely, when there is an error, the system values update to reduce this error. The approach can be described with the following set of equations (the notation used is borrowed from [12], which in turn followed from [11]).

Let A_t be the actual output at time t and A_{t+1} be the actual output on the following time step.

$$A_{t+1} = A_t + \Delta_t \tag{4}$$

where

$$\Delta_t = \alpha(T_t - A_t). \tag{5}$$

Δ_t is the product of a learning rate (α) and the difference between the actual output at t (A_t) and the target output (T_t).

If the target value remains constant, A_t will converge to T_t at the rate determined by α. However, a moving target can cause A_t to oscillate around the target value. In order to dampen the oscillations, an additional variable known as the *momentum* term (μ) can be introduced, transforming the equation to:

$$\Delta_t = \mu\Delta_{t-1} + \alpha(1-\mu)(T_t - A_t). \tag{6}$$

The delta rules expressed above form the basis of the update rule for the MRU threshold. However, as with [12], the target threshold required at each time step is actually unknown and therefore needs to be derived from the data available to the broker. An additional associated variable in the form of a normalized version of the projected resource utilization rate is used, denoted τ. Remembering that a lower θ (close to 0) encourages the purchasing of more reservations, while a higher θ (close to 1) encourages purchasing fewer reservations, τ can be determined:

$$\tau = \frac{reservationsOwned}{summedDemand + 1} \tag{7}$$

where 1 is added to the denominator for cases of no demand.

The rationale for this approach lies with the ultimate aim of the broker to maximize profit through the constant full utilization of the reservations owned, in which case the more expensive on-demand instances would not be purchased and reservations would not go unused. The choice is not without its complications, however. For instance, if the broker owns a relatively large number of reservations, say 100, and the demand for reservations is low, for example 10, the target becomes $\frac{100}{10+1} \approx 9.1$. This is clearly not a suitable target threshold as it exceeds the maximum value of θ considerably. The proposed solution for this involves normalizing the outputted value (see eqrefeqn:tau2) by keeping track of the largest recorded raw target and normalizing the values between 0 and 1. In this particular example, if 9.1 was the largest seen so far, it would be normalized to 1. If a raw target of 10 had been seen in a previous month, it would be normalized to 0.9, *et cetera*. We normalize τ such that:

$$\tau = \frac{\tau - minTarget}{maxTarget - minTarget} \tag{8}$$

TABLE III. HIGHEST RANKING (μ, α) PAIRINGS ACROSS MARKETS

μ	α	Avg. Rank (440 max)
0.7	0.05	400
0.8	0.85	393.75
0.1	0.8	387.25
0.45	0.8	377.5
0.4	0.3	371.25

where $minTarget$ and $maxTarget$ are updated over time to determine the relative value of τ. Then, letting θ_t and θ_{t+1} be the threshold at time t and $t+1$ respectively and substituting in τ as the target value, we derive the following AAT formulation from (4) and (6):

$$\theta_{t+1} = \theta_t + \Delta_t \tag{9}$$

where

$$\Delta_t = \mu\Delta_{t-1} + \alpha(1-\mu)(\tau - \theta_t) \tag{10}$$

and $\Delta_0 = 0$. The three parameter settings must all fall within the range: $0 \leq \tau,\ \alpha,\ \mu \leq 1$.

B. Selecting Robust AAT Parameters

The reader will notice that AAT introduces new variables to the brokerage model. The value of τ is calculated during run-time using (7) and (8). However, the broker must select parameter values for α and μ. Here, we aim to determine AAT parameter settings that work well *out of the box* under a range of market conditions. This configuration should then enable the broker to maximize profit under a range of market conditions, by self-adapting θ over time in response to variation in demand. In this way, the broker no longer needs to determine θ using *a priori* knowledge of the market they are operating in, thus enabling a more *robust* brokerage model.

To determine appropriate AAT values, we trialed a range of values for $0 \leq \alpha, \mu \leq 1$ (at resolution 0.05), in a variety of market conditions. Table III shows the average ranking of pairwise (μ, α) combinations across the full series of trials. We see that $(\mu, \alpha) = (0.70, 0.05)$ consistently performs well and generates the most profit across all markets. Thus, we use these values to configure AAT for the remainder of experiments performed here, and suggest this configuration as suitable for using the AAT brokerage model *out of the box*. We test the robustness of this configuration in each market, to observe:

1) Convergence behavior: does $\theta^{AAT}_{t\to\infty}$ converge to θ_{opt}?
2) Initialization sensitivity: does the starting threshold value, $\theta^{AAT}_{t=0}$, affect the convergence behavior?
3) Profitability: how does AAT compare with the known static θ_{opt} for each market?

Three starting thresholds were tested: $\theta^{AAT}_{t=0} \in \{0, 1, \theta_{opt}\}$. Each experimental configuration was repeated 30 times.

Fig. 6 shows the yearly mean threshold value, θ, generated by AAT in the Recession & Recovery Market. It can be clearly seen that, under each condition, the value of θ quickly converges toward $\theta_{opt} = 0.8$, but equilibrates slightly higher. This demonstrates good convergence behavior and insensitivity to the starting value $\theta^{AAT}_{t=0}$. In other markets, AAT convergence

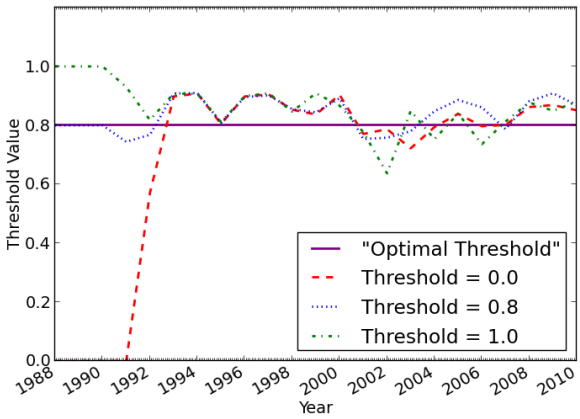

Fig. 6. Yearly mean threshold value, θ, generated by AAT in the Recession & Recovery Market. Under each starting condition, $\theta_{t=0}^{AAT} \in \{0, 0.8, 1\}$, AAT equilibrates near the optimum threshold value, $\theta_{opt} = 0.8$.

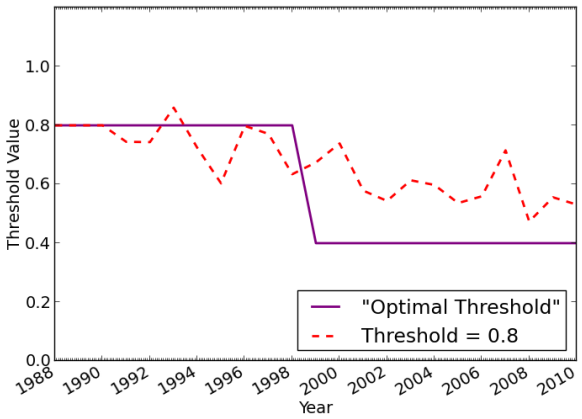

Fig. 7. Market shock from Recession & Recovery to Rapid Growth market.

TABLE IV. PROFITABILITY ($M) OF AAT ACROSS MARKETS

Market	Mean Profit ($M) Using Different Configurations			
	Static θ	AAT		
	$\theta = \theta_{opt}$	$\theta_{t=0}^{AAT} = 0$	$\theta_{t=0}^{AAT} = 1$	$\theta_{t=0}^{AAT} = \theta_{opt}$
Rapid Growth	1.088	1.0765 (-1.06%)	1.0789 (-0.84%)	1.0765 (-1.06%)
Steady Growth	1.377	1.367 (-0.73%)	1.362 (-1.09%)	1.367 (-0.73%)
R & R	1.600	1.610 (+0.63%)	1.594 (-0.38%)	1.614 (+0.88%)
Steady	1.764	1.739 (-1.42%)	1.735 (-1.64%)	1.783 (+1.08%)

is also insensitive to initial conditions (figures not shown, see [16] for more details). However, in other markets, AAT tends to converge to a value $\theta_{t\to\infty}^{AAT} > \theta_{opt}$. Hence, AAT tends to be more *conservative* than the static method, purchasing fewer VM instances than θ_{opt}. Table IV tabulates the profitability of AAT in each market, compared with the profitability of the static threshold, θ_{opt}. We see that, in each market, AAT performs well against the static θ_{opt}, at worst generating 1.64% *less* profit (*Steady* market, $\theta_{t=0}^{AAT} = 1$), and at best generating 1.08% *more* profit (*Steady* market, $\theta_{t=0}^{AAT} = \theta_{opt}$). Since this spread of profits is very close to that achieved by the static θ_{opt}, we can conclude that across all markets, AAT: (1) converges toward the known optimal value θ_{opt}, or a more *conservative* value greater than θ_{opt}; (2) is largely insensitive to initial conditions, $\theta_{t=0}^{AAT}$; and (3) can compete with the known static optimum value, θ_{opt}. Since AAT requires no domain knowledge and no *a priori* optimization in each market, we therefore conclude that AAT is a robust extension to the static thresholding technique introduced by R&C. Although we have shown AAT to be largely insensitive to initialization, as a simple heuristic, we suggest initializing AAT to $\theta_{t=0}^{AAT} = 0.5$. This should minimize the average distance to the market optimum, θ_{opt}, and hence should accelerate time to convergence and increase profit.

C. Market Shocks

Here, we perform a final set of experiments to test the utility of AAT when there is a *market shock*, such that market demand suddenly changes from one profile to another. Market shocks occur in real markets when there is a rapid change in demand, perhaps caused by a new market entrant (e.g., see [17] for a discussion on adapting to market shocks). By testing AAT in shocked markets, we aim to simulate more realistic market dynamics. For these experiments, we use the values $\alpha = 0.45$ and $\mu = 0.55$. These were shown to perform well during a series of preliminary experiments.

Fig. 7 shows threshold values, θ^{AAT}, over time (red dash) in one simulation run of a market that is initially a *Recession & Recovery* market and then *shocked* to become a *Rapid Growth* market. The optimal static threshold value, θ_{opt}, is represented by the purple line. We see that $\theta_{opt} = 0.8$ while the demand profile is *Recession & Recovery* and then falls to $\theta_{opt} = 0.4$ while the demand profile is *Rapid Growth*. Initialized with $\theta_{t=0}^{AAT} = 0.8$, we see θ^{AAT} fluctuate around $\theta_{opt} = 0.8$ during the *Recession & Recovery* market phase, and then decline during the *Rapid Growth* market phase, tending to a value of $\theta \approx 0.5$. This value is greater than $\theta_{opt} = 0.4$, but much lower than the optimum value in the *Recession & Recovery* market. This figure illustrates AAT adapting θ appropriately when the market is shocked. However, in other experiments, AAT is not so well behaved (results not shown, refer to [16]). Overall, we conclude that in markets that are shocked, AAT offers advantages over the static method employed by R&C, which is unable to adapt. Yet, results are preliminary and we believe that AAT should be further refined in order to improve the performance. To achieve this, one method that could be employed is "computational steering" [18]; where a computational system is manually *steered* by a human pilot during run time. Unlike a fully autonomous system that is preconfigured and then left to run in isolation with no further human intervention, a computational steering approach to adaptive thresholding would enable the broker to *steer* the AAT parameters over time as market dynamics change. In this way, computational steering enables human input to the system that is otherwise difficult, or *impossible*, to operationally define, such as a domain expert's *tacit knowledge*, or *intuition*.

The market shock experiments reveal the importance of the early stages of reservation hedging for the broker's overall

performance. As reservations are a long-term (36-month) investment and since the broker cannot see into the future, there is little that can be done in the short term to circumvent a situation where the broker suddenly owns significantly more or less reserved instances than required. The reader should note that R&C's MRU threshold, θ, controls the proportion of months that the broker is prepared to accept an estimated resource deficit. This is calculated on a monthly basis based on a three year history of demand data. Hence, when a market shock occurs, the MRU technique is negatively disrupted as the previous demand data becomes less relevant to future demand forecasts. As a result, R&C's MRU technique becomes weak when the market is shocked. In contrast, AAT attempts to overcome this problem by enabling the broker to adapt the number of reservations purchased depending on the incoming demand, even if it is historically atypical. However, the model is still constrained by R&C's demand estimation routine: that future demand can be directly forecast from historical demand. In future, we would like to try an alternative demand estimation model, such as the statistical model presented in [19].

VII. Conclusion

We have replicated and extended the cloud brokerage simulation model of Rogers & Cliff (R&C) using *CReST*, an open-source, discrete event, cloud data center simulation platform developed at the University of Bristol. To our knowledge, this is the first replication of R&C in the literature and we present our work as validation of their model. However, sensitivity analysis has revealed that R&C's brokerage model is sensitive to configuration parameters, such as: the pricing tariff providers charge for resources, the pricing structure brokers charge their clients, and the effect of noise in the market demand profiles. We present this as evidence that R&C's model requires *modification* before it can be practically used in the real world. To overcome this, we have introduced a novel extension to R&C's model that enables the broker to automatically adapt during run-time to maximize profits, without the broker needing to provide *a priori* knowledge of the market demand or other model parameters. We have demonstrated that this automated adaptive thresholding (AAT) technique is able to converge toward the known optimal value in all markets and that it is robust to initial conditions. We present this as evidence that AAT is a practical, robust extension to R&C's model. We believe this could have practical significance in the real-world market for cloud computing.

Acknowledgment

Thanks to Owen Rogers for detailed discussions of his model implementation and for supplying his demand data to enable replication. Primary financial support for John Cartlidge comes from EPSRC grant number EP/H042644/1.

References

[1] B. Hayes, "Cloud computing," Communications of the ACM - Web Science, vol. 51, no. 7, Jul. 2008, pp. 9–11.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, U.S. Department of Commerce, Tech. Rep. NIST Special Publication 800-145, Sep. 2011.

[3] M. Armbrust et al., "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. EECS-2009-28, Feb. 2009.

[4] O. Rogers and D. Cliff, "A financial brokerage model for cloud computing," Journal of Cloud Computing: Advances, Systems and Applications, vol. 1, no. 2, Apr. 2012, doi:10.1186/2192-113X-1-2.

[5] F. Wu, L. Zhang, and B. A. Huberman, "Truth-telling reservations," Algorithmica, vol. 52, no. 1, 2008, pp. 65–79.

[6] S. H. Clearwater and B. A. Huberman, "Swing options: a mechanism for pricing IT peak demand," in 11th Int. Conf. on Computing in Economics & Finance CEF-2005, Washington, D.C., Jun. 2005. [Online] http://www.hpl.hp.com/research/idl/papers/swings [retrieved: Aug, 2013].

[7] CReST, "CReST - the Cloud Research Simulation Toolkit," [Online] https://sourceforge.net/projects/cloudresearch/ [retrieved: Aug, 2013].

[8] J. Cartlidge and D. Cliff, "Comparison of cloud middleware protocols and subscription network topologies using CReST, the cloud research simulation toolkit," in 3rd Int. Conf. Cloud Computing & Services Science (CLOSER-2013), F. Desprez et al., Eds. Aachen, Germany: SciTePress, May 2013, pp. 58–68.

[9] O. Rogers and D. Cliff, "Forecasting demand for cloud computing resources: An agent-based simulation of a two tiered approach," in 4th Int. Conf. Agents & Artificial Intelligence, vol. 2 - Agents (ICAART-2012), J. Filipe and A. L. N. Fred, Eds. Vilamoura, Algarve, Portugal: SciTePress, Feb. 2012, pp. 106–112.

[10] B. Widrow and J. M. E. Hoff, "Adaptive switching circuits," IRE WESCON Convention Rec., vol. 4, Aug. 1960, pp. 96–104.

[11] D. Cliff and J. Bruten, "Minimal-intelligence agents for bargaining behaviours in market-based environments," Hewlett-Packard Labs., Tech. Rep. HPL-97-91, Aug. 1997. [Online] http://www.hpl.hp.com/techreports/97/HPL-97-91.pdf [retrieved: Aug, 2013].

[12] J. Cartlidge and D. Ait-Boudaoud, "Autonomous virulence adaptation improves coevolutionary optimisation," IEEE Transactions on Evolutionary Computation, vol. 15, no. 2, 2011, pp. 215–229.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations, D. E. Rumelhart and J. L. McClelland, Eds. MIT Press, 1986, pp. 318–362.

[14] S. W. Wilson, "ZCS: A zeroth level classifier system," Evolutionary Computation, vol. 2, no. 1, 1994, pp. 1–18.

[15] ——, "Classifier fitness based on accuracy," Evolutionary Computation, vol. 3, no. 2, 1995, pp. 149–175.

[16] P. J. Clamp, "Pricing the cloud: An investigation into financial brokerage for cloud computing," Master's thesis, Dep. Comp. Sci., Univ. Bristol, UK, July 2013.

[17] S. Stotter, J. Cartlidge, and D. Cliff, "Exploring assignment-adaptive (ASAD) trading agents in financial market experiments," in 5th Int. Conf. on Agents & Artificial Intelligence, Vol. 1 - Agents (ICAART-2013), J. Filipe and A. L. N. Fred, Eds. Barcelona, Portugal: SciTePress, Feb. 2013, pp. 77–88.

[18] S. Bullock, J. Cartlidge, and M. Thompson, "Prospects for computational steering of evolutionary computation," in Workshop Proc. 8th Int. Conf. Artif. Life (ALife-VIII), E. Bilotta et al., Eds. Sydney, Australia: MIT Press, Dec. 2002, pp. 131–137. [Online] http://eprints.ecs.soton.ac.uk/11459/1/Prospects.pdf [retrieved: Aug, 2013].

[19] J. Cartlidge and S. Phelps, "Estimating demand for dynamic pricing in electronic markets," GSTF International Journal on Computing (JoC), vol. 1, no. 2, 2011, pp. 128–133.