Dynamics of Resource Closure Operators

Alva L. Couch and Marc Chiarini

Tufts University, Medford, Massachusetts, USA
couch@cs.tufts.edu, marc.chiarini@tufts.edu

Abstract. We propose a framework for managing resources via convergent operators. Operators represent their need for a resource to a designated *resource closure operator* that manages the resource. We evaluate a specific design for a resource closure operator by simulation and demonstrate that the operator achieves a near-optimal balance between cost and value without using any model of the relationship between resources and behavior. Instead, the resource operator relies upon its control of the resource to perform experiments and react to their results. These experiments allow the operator to be highly adaptive to change and unexpected contingencies.

1 Introduction

A *convergent operator*[1, 2] is a process that acts upon a network to achieve a management objective. Each operator has several properties, including:

1. *awareness* (or "knowledge"): The operator has a mechanism by which it can determine if its goals have been achieved.
2. *idempotence* (or "convergence"): If its goals have been achieved and they remain so, the operator effects no changes in the network.

Each operator functions like an independent autonomic control loop: performing monitoring, planning, and system changes at every invocation. The function of a control loop is achieved by repeating one operator over time, either at regular time intervals or by chance[3]. Convergent operators are applied periodically to "immunize" a system against performance degradation[2, 1, 4]. This basic philosophy is exemplified by the CfEngine system for configuration management[5, 6]. Prior work shows that autonomic computing can be "approximated" by a collection of convergent operators applied repeatedly at random[7]. Our operator model is motivated by the "maintenance theorem" of Burgess[3], which demonstrates that autonomous entities can converge to an emergent fixed-point without coordination or coercion.

In this paper, we show that Burgess' maintenance theorem can be applied to resource management, by defining appropriate operators and studying their behavior. This work is a continuation of our prior work on convergent operators[8]. Two convergent operators are *consistent* with one another if probabilistic, repeated application of both of them leads to a network state that achieves both of their management objectives simultaneously, and *inconsistent* otherwise. We

demonstrated that consistency is best viewed as a dynamic, emergent property of an operator system, rather than something that can be analyzed statically. We also introduced the idea of a *closure operator* that acts on a system within well-defined limits and is self-healing only within a very narrowly defined domain of responsibility[9]. A closure is a self-managing part of an otherwise open system[10–12]. It is called a closure partly because it acts something like a closure in a programming language: it is an island of determinism in a perhaps unpredictable overall system. In this paper, we discuss the design of a specific closure operator for resource management.

This paper is motivated by current challenges to autonomic control models of system management. At Hot Autonomic Computing 2008 (HotAC08), one of the grand challenge problems discussed was that of composing several autonomic control loops with perhaps different objectives[13]. It was accepted that without further constraints, two control loops controlling the same behavioral domain cannot be used at the same time with predictable effects. Thus, a definition of the concept of loop composition remains unclear. Another grand challenge problem was to determine complete and accurate models of server performance. Without these models, traditional control theory (as presented in [14]) is significantly hampered in guaranteeing acceptable response.

Our research responds to these challenges with a new model of resource management that utilizes closures with partial information to replace closed loops with complete information. We describe the design of a closure operator that manages a one-dimensional resource parameter and balances demand (value) against available resources (cost). We demonstrate that an efficient balance between cost and value can be achieved without modeling system dynamics, even in cases where demand functions change suddenly. The result is a highly adaptive management strategy that utilizes minimal information about the environment to maximize its objective function.

The organization of this paper is as follows. We begin by introducing a scalar-value resource-management problem and our architecture and assumptions. We then define our algorithm and demonstrate its properties via simulation. Finally, we discuss limitations and directions for future work.

2 A Scalar-Value Management Problem

Our control model is depicted in Figure 1. A system to be managed outputs one value: a *performance measure* P. It is managed by controlling a *resource parameter* R, which is assumed to (perhaps indirectly) determine P, though there are other determining factors for P, including system load X (which is known to exist but is not considered in our model). P might represent response time, mean time in system, etc., while R might represent memory, server instances, etc. In this study, the goal of management is to control R so that the value of P balances with the cost of R.

This control model is motivated by concerns that arise in the context of Business-Driven Information Technology (BDIM)[15, 16]. To truly optimize an

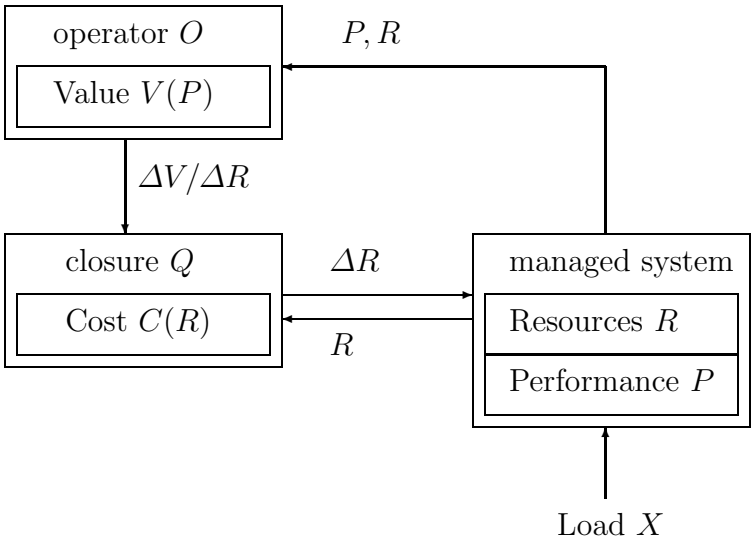

Fig. 1. Operator interaction occurs through a resource closure.

enterprise, it is not enough to maximize delivered value; one must also balance cost and value. Thus our objective function is $V - C$, the difference between value V and cost C. We aim to maximize $V - C$ based upon constraints on resources R.

In our model, R is controlled via an intermediary *resource closure operator* Q that accepts recommendations from (distributed) operators O and then recommends changes to be made in R. Each recommendation is a quantity $\Delta V/\Delta R$ that represents the local rate of change in value to the operator in question. Q computes the sum of value estimates from each source to get an overall $\Delta V/\Delta R$ representing the total potential gain from increasing R. The closure then combines this with a local concept of the cost $C(R)$ of the resource to compute a net reward difference $\Delta(V-C)/\Delta R$. If this difference is positive, Q recommends a positive increment ΔR in resources, while if it is negative, a negative ΔR is recommended instead. Q decides upon the magnitude of ΔR independently of the value of $\Delta(V - C)/\Delta R$. For this paper, that magnitude is a constant value that differs only in sign over time.

While cost is a function of available resources ($C = C(R)$), value is a function of performance ($V = V(P)$). We think of P as average response time to queries. P has an "open" and unknown relationship with R. It is not a simple function of R, but rather, depends upon other unknowns (including, e.g., an unknown load X). We might notate P as $P(R, X)$ where X is some unknown vector of load attributes. Our model does not consider these influences: each operator models value inaccurately as a function of P alone. Inaccuracies in the model of P are mitigated via *agility* in both error detection and correction. This is exactly the approach advocated by Burgess[3].

Our model makes a bare minimum of assumptions. It assumes that $P(R, X)$ is simply decreasing in R for constant X, that is, $P(R + \Delta R, X) < P(R, X)$ when $\Delta R > 0$. In other words, more resources improve performance (by decreasing response time). If R can take real values, this implies that P is continuous in R, but R often assumes only a discrete set of numeric values (e.g., integers).

Other than this the model makes no general assumptions about the time-varying behavior of P. It does not assume that P changes predictably in response to changes in X, or even that P remains the same function between time t and time $t + 1$. It might remain the same or it might change.[1]

We have left behind the assumptions of classical control theory. Most applications of control theory require knowledge of $P(R, X)$ itself, and we do not presume such knowledge, because there may be other active management processes that modify the system and thus influence X (and P). We only control one factor influencing P, which is R. We sidestep knowledge of X and replace that knowledge with controlled experimentation in real time. Our overall model embodies an *open world assumption*, while most control theory operates in a *closed world* in which all influences on a system are known in advance.

2.1 Composition

Using our model, composing value estimates from multiple operators is straightforward and trivial, while composing typical autonomic control-loops is considered difficult or perhaps impossible, because of lack of ability to predict interactions between the loops. We have not simplified the problem; our model only moves the difficulty to a different place. The model describes a complex *dynamical system* with seemingly unpredictable behavior. While composition of multiple management loops is made easy by the architecture, "understanding" such a system is made more difficult by its nature. The precise behavior of our systems *cannot* be predicted, and approximate bounds on behavior must suffice as aids to understanding.

2.2 Observability

The assumptions we have made so far seem straightforward but have profound effects. The fact that the performance function can change without notice (e.g., by replacing the whole machine that performs the service) means that information collected far in the past may well describe a *different* P from the present one. To reasonably assure that we are estimating "the current P", only recent information can be safely used, and even that can be incorrect if P has just changed.

Although there is no rule against operators collecting information over long time periods, this information need not be useful under our assumptions. Thus we limit the operators O and Q to using estimates for P and R from the last few

[1] The effectiveness of our operators requires that it does not change very often, but frequency of change is not required to be limited in order to *apply* the strategies.

time steps only (e.g., ten previous operator invocations). While there is clear value in doing so, we do not attempt to detect whether X or the function P itself has changed. The simulations presented below demonstrate that we do not need to detect such changes in order to obtain useful results. In fact, we must limit our observation window to decrease the time in which the model responds to dramatic changes in P.

2.3 Estimating $\Delta V/\Delta R$

The convergent operators O provide some estimate of $\Delta V/\Delta R$ by fitting recent measurements of $V(P)$ to recent measurements of R via a linear model with least-squares fit. The slope of the R coefficient in this model serves as an estimate of $\Delta V/\Delta R$. Each operator has one tunable parameter, which is the amount of history it utilizes in estimating $\Delta V/\Delta R$. If this amount of history is small, it reacts quickly, while large amounts of history make response to sudden changes more gradual, and serve as a kind of noise filter.

This kind of sampling is very different from – and perhaps even contrary to – the kind of sampling currently done in Cfengine. Cfengine attempts to smooth out errors in measurement via moving averages. By contrast, we utilize un-smoothed measurements and expect inaccuracies and noise. The reason our strategy still works is that we filter the results in a different way, inside the resource closure Q.

Note that this is not a trial-and-error approach, nor do we advocate that approach. Our experiments indicate that the validity of this approach depends very much on how Q reacts to information from O. It is best to think of Q as performing controlled experiments. Each change in R is an experiment whose results are available during the next application cycle. Q's goal in this experimentation is to move R nearer to an optimum value, on the assumption that the estimates of $\Delta(V - C)/\Delta R$ become more accurate as the optimum value is approached. If one violates this pattern, the algorithm becomes divergent and of no practical use. We performed many experiments in which Q failed to condition its data sufficiently to allow managing R.

2.4 The Resource Closure

The resource closure Q sums the various $\Delta V/\Delta R$ estimates it receives to get a total estimate. It then subtracts its own estimate of $\Delta C/\Delta R$ to get a total estimate of $\Delta(V-C)/\Delta R$. This is the hypothetical gain in the objective function $V - C$ for unit change in R. If this is positive, the closure increases R by some ΔR, while if it is negative, it decreases R by some ΔR. The magnitude $|\Delta R|$ of the increment is the closure's only tunable parameter. If $|\Delta R|$ is too small, then convergence speed is too slow and the system undershoots behavioral goals, while if it is too large, then it overshoots goals and oscillates around the optimum value while never reaching it.

2.5 Perturbations And Seeding

The operators as defined above need data in order to function, creating a "chicken-and-egg' problem if the system is initially stable. If R is initially held constant, then there will never be enough data to estimate any $\Delta V/\Delta R$, because this requires that R is changing. To solve this problem, the parameter closure Q "seeds" the incoming data for the operators by incrementing the values of R by a fixed ΔR whenever operators seem to be in balance. In this situation, Q increments R (unless R is maximal, in which case it decrements), to create experimental data from which the other operators compute their value functions.

3 Simulation Results

To study the behavior of our proposed system, we wrote a custom simulator in the statistical environment R[17], and explored the design of Q through several simulation experiments. The basis for our experiments is the following simulated system, containing an environment, one management operator, and one closure operator.

The environment has the properties that:

1. $X(t) = 1000 \sin((t/p) * 2\pi) + 1000 + e(0, \sigma)$, time-varying periodic load between 1000 and 2000 with Gaussian error $e(0, \sigma)$.
2. $P(R, X) = X/R$, analogous to response time in a server farm of R servers, with perfect efficiency.

X represents load arriving from outside the environment, while P represents behavior as a function of load X and resources R. The formulae for these properties are not known to any operator, though each operator can observe a small number of pairs P, R to estimate $P(R)$.

There is one operator[2] O, with:

1. $V(P) = 200 - P$ (diminishing returns in proportion to increased response time).
2. w measurements (its *measurement window*), used to compute its least-squares estimate of $\Delta V/\Delta R$.

There is one resource closure Q, with:

1. $C(R) = R$ (analogous to adding same-cost increments of resources).
2. $|\Delta R| = d$ (constant increment magnitude).
3. $R \in [1, 1000]$ (finite resource bound).
4. $R_{\text{initial}} = 50$ (modest amount of resources allocated to reduce startup settling time).

[2] Since operators compose by adding their outputs, it is sufficient to show that one such function is manageable.

$X(t)$ and $R(t)$ independently vary with time, and cause variation with time in $P(R, X)$, $C(R)$, and $V(P)$.

In the above, there are four model parameters we can vary in simulation:

1. σ represents the standard deviation of a Gaussian error term $e(0, \sigma)$, representing a situation in which measurement errors are normally distributed.
2. p represents the period of the periodic load function, which affects the optimal increment d to use in updating R.
3. d represents the constant addend that Q uses to update R.
4. w represents the number of samples in O's estimate of the rate of change of V.

This instance was chosen for study because:

1. It has an easily computable theoretical best value for R, which is $R_{\text{best}}(X) = \sqrt{X}$.
2. The optimal solution does not reach the boundaries of R's range.
3. There is exactly one local maximum so hill-climbing is guaranteed to converge to that maximum.
4. P, V, C, and X are minimally complex but provide interesting behavior[3]

Note that the operator O estimates $\Delta V/\Delta R$ by assuming that there is a linear relationship between V and R, but that the actual relationship, unknown to O, is $V = 200 - P = 200 - X/R$, which is hyperbolic. Our tests indicate that this inaccuracy does not matter at all for our purposes, and we observed no difference between using a linear model and the more accurate hyperbolic model in simulation. Part of the reason is that neither the linear nor the inverse-linear model accounts for X, so that both are *equally* inaccurate.

There are several roles of Q in managing the system. One effect of a fixed $|\Delta R|$ is to quantize the values R can take to a set of finitely many achievable values, thus controlling the data from which $\Delta V/\Delta R$ is computed. Q enforces this set of values, and keeps values within feasible range. Quantization reduces errors in estimates of $\Delta V/\Delta R$ by keeping ΔR from becoming too small.

In early experiments, we tried to increment R by $\Delta R/\Delta(V - C)$ (for one unit improvement in cost), but this showed divergent behavior due to errors in estimating $\Delta(V - C)/\Delta R$. The behavior did not improve when substituting any constant multiple of the addend. Clearly, exposing a parameter to raw increment recommendations was "too risky". Thus we chose to utilize a constant increment instead.

3.1 Simulating The Model

In the simulation, we assume that all operators are invoked at each time step, followed by invoking Q on their results. We compared the performance of the system in each test case to the best-case performance in which the resource function tracks the theoretical optimum (with no noise).

[3] Note that if V and C are both linear in R, then either the system saturates at one end of R's range, or is in balance for every R if $\Delta V/\Delta R = \Delta C/\Delta R$.

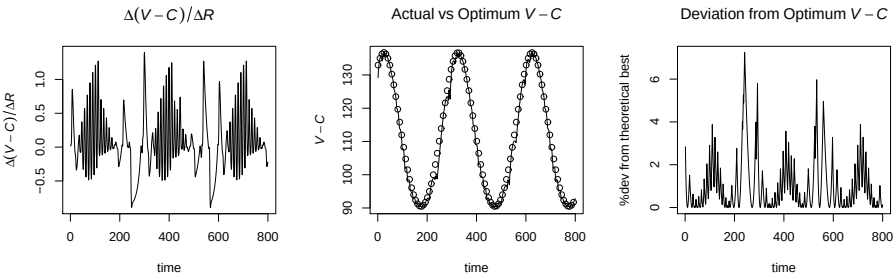

Fig. 2. Input to Q is seemingly unpredictable and random(left), but the output of the system is surprisingly close to the theoretical best performance, shown as circles in the middle plot. The percent difference from theoretical best performance is small (right).

Results of an example simulation are shown in Figure 2. In this simulation, $p = 300$, $d = 1$, $w = 10$, and $\sigma = 0$; in all figures in this paper, the first 200 steps are omitted to allow the system to settle into periodic behavior. On the left is the raw input to Q, expressed as $\Delta(V - C)/\Delta R$. This is often unpredictable, swinging from positive to negative as the naïve estimator of $\Delta V/\Delta C$ makes mistakes due to lack of knowledge of X. These mistakes are mitigated by ignoring the magnitude of $\Delta(V - C)/\Delta R$ and considering only its sign. The agile way in which the system corrects estimation errors allows it to track well the theoretical target (shown in the middle, as a series of circles). The percent error between theoretical and simulated behavior is surprisingly small (right).

The reader might assume incorrectly at this point that Q is merely receiving information from O and acting upon that received information. It is better to think of Q as "constructing experiments" to which O reacts by receiving and forwarding experimental results. While the results of experiments are somewhat random[4], Q's structure assures that the statistical behavior (e.g., how well the system tracks an ideal performance goal) is less random and more predictable.

To understand Q's role in constructing experiments, consider the unexpected downturn with X increasing in Figure 2, at about $t = 250$. The reason the downturn occurs is that X increases quickly enough that $\Delta(V - C)/\Delta R$ becomes negative, suggesting that R should be decreased when in fact it should be increased. The *next* experiment of Q decreases R and tries again, with the result that $\Delta(V - C)/\Delta R$ decreases more quickly. This allows Q to correct its mistake and proceed in the direction of best value, *without becoming aware* that changes in a latent independent variable X were involved in its mistake.

3.2 Effect Of Increment Magnitude

The effect of changing the increment d in the simulation is substantial. Consider the effect of $d = 1, 3, 5$ upon the original model (Figure 3). In the figure, the circles represent theoretically optimal behavior. In this experiment, $p = 100$,

[4] Again, due to lack of ability to observe determinism.

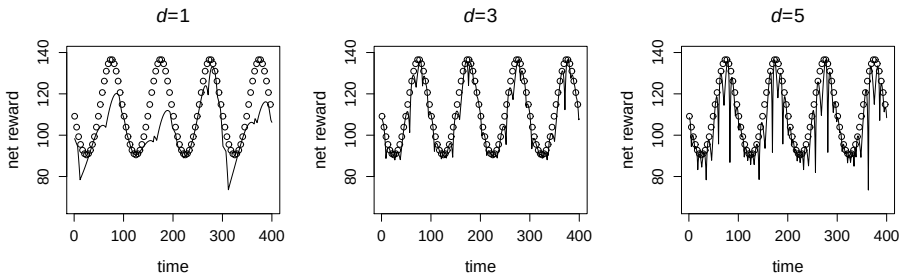

Fig. 3. The effect of different choices for ΔR includes overshoot (right) and lack of tracking (left).

period	$d=1$	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$	$d=7$	$d=8$
50	96.6	92.6	91.1	94.9	93.7	92.3	90.7	86.8
100	**93.4**	97.4	**97.4**	96.6	**95.1**	92.9	90.5	80.4
150	93.4	98.4	97.9	96.9	95.5	92.7	89.7	80.7
200	97.3	98.7	98.2	97.1	95.3	91.8	89.7	81.7
250	98.3	98.9	98.3	97.0	95.3	92.1	89.1	81.1
300	98.9	99.1	98.3	97.1	95.1	91.9	89.2	80.6
350	99.2	99.1	98.2	96.9	95.0	91.8	89.4	79.4
400	99.3	99.2	98.4	96.8	95.0	91.7	89.2	81.0
450	99.5	99.2	98.2	97.0	94.5	91.8	89.3	81.5
500	99.6	99.2	98.3	96.9	94.6	91.8	89.2	80.7

Table 1. Efficiencies of combinations of increment and period indicate that increment is a critical parameter of the algorithm. Efficiencies for the conditions in Figure 3 are shown in bold. For all tests in this table, $\sigma = 0$ and $w = 10$.

$w = 10$, and $\sigma = 0$. Note that $d = 1$ causes convergence to be too slow and remain lower than optimal (undershoot), while $d = 5$ causes repeated jumps around the optimum value (overshoot). $d = 3$ is approximately appropriate for $p = 100$, which controls how quickly X changes. The best choice for d depends upon the maximum of $\Delta X/\Delta t$ (which is unknown to and not discoverable by Q).

We compared the efficiency of several combinations of increment and period, to understand their relationship (Table 1). The efficiency of each run is defined as the ratio of sums of $V - C$ to the theoretical best $V - C$:

$$E = \frac{\Sigma_i(V_i - C_i)}{\Sigma_i(V_i^{\text{best}} - C_i^{\text{best}})}$$

(which is the same as the ratio of their averages), where $V_i^{\text{best}} - C_i^{\text{best}}$ is the maximum theoretical payoff. The results clearly show that there is a best d for every p in achieving maximum efficiency. We conclude that d is a critical parameter and a good candidate for dynamic tuning.

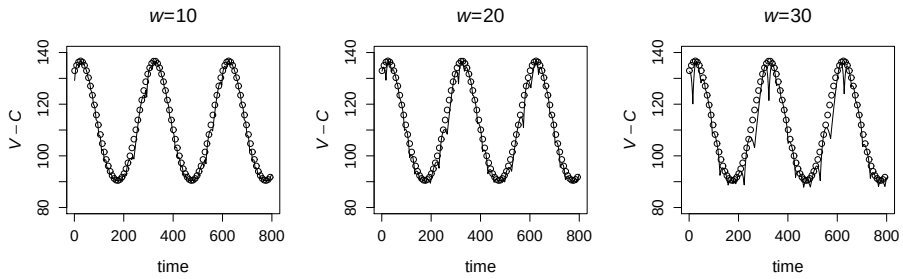

Fig. 4. The effect of increasing window size is to magnify errors in estimation.

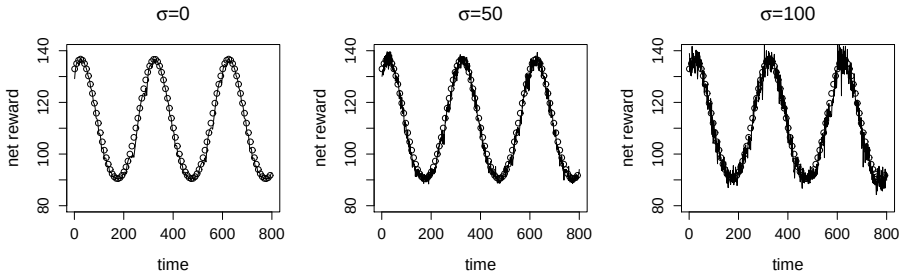

Fig. 5. The effect of greater amounts of noise is to increase convergence time. Notice the increase in oscillations around the optimal as noise increases.

3.3 Effect Of Window Size

Our next experiment was to compare the effects of various window sizes $w = 10, 20, 30$ with period $p = 300$, increment size $d = 1$, and noise magnitude $\sigma = 0$ (Figure 4). Note that as window size increases, the tracking of the ideal value is less accurate, and that larger window sizes *magnify errors in estimation* rather than lending accuracy by smoothing. This is the opposite of what one might expect.

To understand this effect, consider that w determines the number of observations involved in estimating value, where at any one time some of these may be in error. Estimating value based upon erroneous data leads to erroneous conclusions, such that a larger w provides the potential for each erroneous observation to have a greater impact.

3.4 Effect Of Noise

Noise has the effect of making the estimates $\Delta V/\Delta R$ inaccurate, and thus affects convergence time. We simulated the system described above with varying values of σ. Figure 5 exhibits three such experiments, corresponding to $\sigma = 0, 50, 100$. In this experiment, $p = 300$, $d = 1$, and $w = 10$. The circles represent optimum performance in the absence of noise.

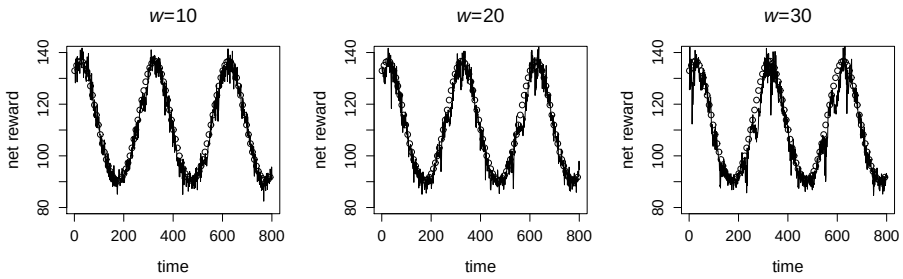

Fig. 6. The effect of different sizes for an operator's history window. A value of w that is too large interacts with the inaccuracy of the model to make overall performance less responsive.

3.5 Effect Of Window On Noise

We next repeated the experiment in Section 3.3 but added a constant magnitude of Gaussian noise. We thought that larger windows would be beneficial, but actually smaller is always better (Figure 6). In the figure, noise magnitude $\sigma = 100$, period $p = 300$, increment size $d = 1$, and window $w = 10, 20, 30$.

The efficiencies of several variations are shown in Table 2. Again, the efficiency of each run is the ratio of the sum of payoffs over time to the sum of best payoffs. The first row contains data for zero noise. Larger windows always lead to worse efficiency. Thus we conclude that small values of w (between 3 and 10) are appropriate.

σ	$w = 5$	$w = 10$	$w = 15$	$w = 20$	$w = 25$	$w = 30$
0	99.1	**98.9**	98.7	**98.5**	98.1	**97.6**
25	99.3	98.9	98.7	98.5	98.0	97.6
50	99.4	99.0	98.7	98.5	98.0	97.6
75	99.3	98.9	98.6	98.1	98.0	97.5
100	98.9	**98.8**	98.5	**98.2**	97.6	**97.6**
125	99.0	99.1	98.5	98.2	97.7	97.5
150	99.1	99.2	98.5	98.1	97.9	97.6

Table 2. Efficiencies of combinations of window and noise show counter-intuitively that response to noise in input is not improved by sampling in larger windows. Efficiencies for Figures 4 and 6 are shown in bold. For all tests in this table, $p = 300$ and $d = 1$.

4 Limitations

Our algorithm has several theoretical limitations to its applicability. It is a hill-climbing algorithm that will always converge to a *local* maximum value, but is

not guaranteed to converge to a *global* maximum value unless there is only one global maximum. It always seeks a global maximum only if there is exactly one local maximum for $\Sigma_i(V_i) - C$, which is true only for the values of R for which the finite difference $\Delta(\Sigma_i V_i - C)/\Delta R$ (which is a function of R) is 0. We must use difference equations to describe this system because derivatives are not always meaningful; the legal values of R can be discrete.

By the assumptions above, the finite differences between values of $\Sigma_i V_i$ and C for values of R near its current value are always strictly greater than 0; we notate these as $\Delta(\Sigma_i V_i)/\Delta R$ and $\Delta C/\Delta R$. Because of this, the combined difference function $\Delta(\Sigma_i V_i - C)/\Delta R$ is zero (the system is in balance) only when $\Delta(\Sigma_i V_i)/\Delta R$ and $\Delta C/\Delta R$ are equal. The only way to assure that there is only one local maximum is for $\Delta(\Sigma_i V_i)/\Delta R$ to equal $\Delta C/\Delta R$ for at most one value of R.[5]

There are only two ways in which this can happen: by a down-crossing, during which value initially changes faster than cost with respect to R and then proceeds to change more slowly than cost, or by the opposite event of an up-crossing. Any value difference function that crosses the cost difference function in the opposite direction from another has the potential to add an extra local maximum. The difference function of the sum of value difference functions that all cross the cost difference function in the same direction also crosses in that direction and is guaranteed to have a unique stable point.

The algorithm also only works if the V_i are never constant over an interval of P, and P is never constant over any interval of R. In other words, the system handles what might be called "highly dynamic" situations, but stops at a non-optimum value of R if V remains constant near that value.

Thus the assumptions we outlined at the beginning are necessary rather than just sufficient, which can make it difficult to design appropriate cost and value functions for practical situations. We believe that handling value functions with regions of constant value requires a different approach, which we plan to discuss in later work.

5 Future Work

Several open issues remain for future work.

First, the algorithms for adjusting d optimally have not been determined, and would be required in order to create a practical implementation. Our simulation data suggests possible solutions, but these have not been evaluated fully.

Second, the efficiency calculations we utilized to evaluate the algorithm are not available if the solution is deployed in the real world. Some way to estimate the efficiency of a real-world solution is needed.

Third, the system is easily generalized to the case in which R is a vector rather than a scalar. X can become a vector without modifying anything in the basic model, and different operators can even have varying *concepts* of X. If R

[5] If these are never equal, then the maximum is either the highest or lowest value of R.

is a vector, then $\Delta V/\Delta R$ becomes a gradient with respect to the components of R, and the parameter closure determines a *direction* in which to move one unit (ΔR) in a quantized R-space rather than a sign of movement in one-space. Much will be determined by exactly how we *quantize* R-space, and this design problem seems difficult at present.

Fourth, there is an intimate relationship between this operator theory and Burgess's promise theory(as discussed in[7]) that is yet to be fully understood. A promise is a non-binding statement of intent. The value judgments communicated to Q are promises to Q, but whether and how one could reason about that information remains unclear, especially if there is coupling between services and if agents might be untruthful about parameter values to their private advantage.

Fifth, this model only handles simply increasing functions. Step-functions and functions with constant-value regions are common in service-level agreements, and there are ways of transforming any such function into a function conforming to our model. We have not studied this in enough detail to comment, though some techniques for approximating step functions with continuous functions seem relevant.

6 Conclusions

We have demonstrated that a model for managing the resources available to a localized service can base decisions on comparing local cost with distributed (and possibly disparate) concepts of value. The model converges to the balance point between value and cost without utilizing any knowledge of the relationship between performance and resources. Instead, success is based upon physical invariants of the environment (e.g., persistence of conditions) that allow the closure operator to statistically discover near-optimal resource levels through repeated experiment. The model is very flexible and requires a minimum of assumptions. One can add and remove at will, operators with different concepts of value. The performance function P can change unexpectedly with quick adaptation to a new optimum. There is a predictable relationship between the rate at which load varies and the best increment to be utilized by Q, although long windows of observation seem to hurt performance in all cases.

This paper centers upon a very simple result, but it opens a Pandora's box of possibilities. It is possible to use distributed concepts of demand and reasoning in concert with a centralized concept of service, and make sense of the global optimum. Convergence of this approach depends only upon the ability of management software to perform controlled experiments, and not upon any model assumptions. This is a first step toward distributed management with no centralized authority.

References

1. Burgess, M.: Configurable immunity for evolving human-computer systems. Science of Computer Programming **51** (2004) 197

2. Burgess, M.: Computer immunology. Proceedings of the Twelth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA) (1998) 283
3. Burgess, M.: On the theory of system administration. Science of Computer Programming **49** (2003) 1
4. Burgess, M.: Cfengine as a component of computer immune-systems. Proceedings of the Norwegian Conference on Informatics (1998)
5. Burgess, M.: A site configuration engine. Computing Systems **8**(2) (1995) 309–337
6. Burgess, M., Ralston, R.: Distributed resource administration using cfengine. Softw., Pract. Exper. **27**(9) (1997) 1083–1101
7. Burgess, M., Couch, A.: Autonomic computing approximated by fixed-point promises. In: Proceedings of the First IEEE International Workshop on Modeling Autonomic Communication Environments (MACE), Multicon Verlag (2006) 197–222
8. Couch, A.L., Chiarini, M.: Dynamic consistency analysis for convergent operators. In: AIMS '08: Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security, Berlin, Heidelberg, Springer-Verlag (2008) 148–161
9. Couch, A.L., Chiarini, M.: A theory of closure operators. In: AIMS '08: Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security, Berlin, Heidelberg, Springer-Verlag (2008) 162–174
10. Couch, A., Hart, J., Idhaw, E.G., Kallas, D.: Seeking closure in an open world: A behavioral agent approach to configuration management. In: LISA '03: Proceedings of the 17th USENIX conference on System administration, Berkeley, CA, USA, USENIX (2003) 125–148
11. Schwartzberg, S., Couch, A.: Experience implementing a web service closure. In: LISA '04: Proceedings of the 18th USENIX conference on System administration, Berkeley, CA, USA, USENIX (2004) 213–230
12. Wu, N., Couch, A.: Experience implementing an ip address closure. In: LISA '06: Proceedings of the 20th USENIX conference on System administration, Berkeley, CA, USA, USENIX (2006) 119–130
13. Couch, A.: Summary of the third workshop on hot topics in autonomic computing (HotAC III). ;login: Magazine **33**(5) (2008) 112–113
14. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. John Wiley & Sons (2004)
15. Sauve, J., Moura, A., Sampaio, M., Jornada, J., Radziuk, E.: An introductory overview and survey of Business-Driven IT management. In: The First IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM). (2006) 1–10
16. Moura, A., Sauve, J., Bartolini, C.: Research challenges of Business-Driven IT management. In: The Second IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM). (2007) 19–28
17. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2008) ISBN 3-900051-07-0.