Designing and Analyzing the Phoenix Planner with Models of Interactions Between Phoenix Agents and the Phoenix Environment.

Paul R. Cohen
COINS Technical Report 90-22

Experimental Knowledge Systems Laboratory
Department of Computer and Information Science
University of Massachusetts, Amherst, MA 01003

Acknowledgments:
This work was supported by ONR Contract N00014-88-K-0009. It was conducted during the author's leave at the Thayer School of Engineering, Dartmouth College. Many people have contributed to this work, especially the members of the Phoenix project: David Hart, Adele Howe, David Westbrook, Scott Anderson, Michael Greenberg, Paul Silvey, Dorothy Mammen, Evan Smith, and Jerry Powell.

Abstract

This paper illustrates how aspects of the design of a planner can be derived from a formal model of the planner's environment and the desired planner behaviors. Specifically, I show how the order of execution of multiple fire-fighting plans is determined by a model of the dynamics of the Phoenix environment. More generally, I introduce an *ecological* approach to the design and analysis of intelligent agents. Some tenets of this approach are familiar; for example, the behavior of an agent arises from its interactions with its environment (e.g., [12, 11, 6, 1, 8]); and some agent designs are better adapted to particular environments than others (e.g., [9, 7, 3, 10]). My purpose here is not to discuss these foundations, but to demonstrate how models of the interactions between an agent and its environment can facilitate design and analysis.

1. An Ecological View

Our goal is to understand the functional relationships between three complex structures: the architecture and knowledge of agents, the structure and dynamics of environments, and the behaviors that result from the interactions between agents and their environments (Fig. 1). Borrowing from the literature on animal behavior, we call these relationships the *behavioral ecology* of an agent.

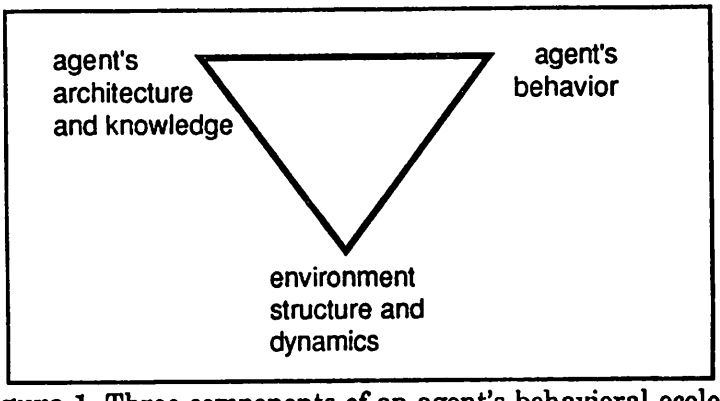

Figure 1. Three components of an agent's behavioral ecology

The terms *agent*, *architecture*, *environment*, and *behavior* are open to interpretation. Without implying that our interpretations are consensual, our view is that agents sense their environments and decide autonomously how to act, and that these actions, moderated by the environment over time, produce behavior. The agent's architecture is a collection of sensors, effectors, and internal data structures and processes. The following section illustrates the design just one aspect of the architecture of Phoenix agents—the decision about the order in which to execute several plans. Our analysis of the current agent architecture, and redesign based on this analysis, is an ongoing project.

The ecological view in Figure 1 suggests six research activities that AI researchers currently engage in or would like to engage in:

Environment assessment: Determining which aspects of the environment must be represented in a model for design and analysis (see Sec. 2.1).

Modelling: Formally specifying the functional relationships from which to predict behavior, given the architecture and environment of an agent (see Sec. 2.2).

Design. Inventing or adapting architectures that are predicted to behave as desired in particular environments (see Sec. 2.3).

Prediction: Inferring from the functional relationships in a model how behavior will be affected by changing the architecture of the agent or its environment (see Sec. 3).

Explanation: Finding the source of incorrect predictions in a model, and revising the model, when unexpected behaviors emerge from the interactions between an agent and its environment (see Sec. 3)

Generalization: Whenever we predict the behavior of one agent in one environment, we should ideally be predicting similar behaviors for agents with related architectures in related environments. In other words, our models should generalize over architectures, environmental conditions and behaviors (see Sec. 3).

In the following sections I will illustrate each of these activities.

2 A Design Problem in Phoenix

The Phoenix project provides many opportunities to explore and develop the ecological view [4]. Phoenix is an environment—a simulation of forest fires—and a collection of simulated agents. Many factors that affect forest fires also affect the fires in the Phoenix environment, including wind speed and direction, elevation gradients, fire temperature and flame height, ground cover, and natural and artificial boundaries such as rivers and roads. Fires, which are implemented as cellular automata, "burn" an array that represents the topographical features of Yellowstone National Park. Nontopographical factors (e.g., weather) are set and changed manually, or randomly, or by prespecified scripts. Phoenix agents contain fires by cutting fireline around them. Several agents are usually required to contain a fire, so they must coordinate or be coordinated. Currently, a single fireboss agent directs several semi-autonomous bulldozer agents, which plan individually how to carry out the directions.

Although Phoenix agents have been designed and implemented, we need to review some design decisions and commit to others which have been pending. Because the design of the current Phoenix agents predates the ecological view outlined above, many design decisions are based on informal characterizations of the Phoenix environment, and so are difficult to analyze (see Sec. 3.)

The current Phoenix agent architecture includes sensors, effectors, reflexes, and a cognitive scheduler. Reflexes respond immediately to situations such as encroaching fire. The cognitive scheduler coordinates all the agent's activities except its reflexes,

including selecting skeletal plans, expanding plans into subplans, assigning appropriate execution methods to actions, monitoring, and fixing plans when actions cannot be executed.

The current cognitive scheduler is rudimentary. It schedules activities in the order that they are generated unless they have high priority, in which case they are done earlier. Unless fires are prioritized, the cognitive scheduler attends to them in the order in which they are reported. The following analysis shows that this is not the best strategy and shows how to prioritize fires to get the best performance.

2.1 Environment assessment.

Environment assessment is the informal process of deciding which aspects of an environment have important effects on particular behaviors, which can be safely left out, and which have unknown effects. The behavior that interests me here is the determination by the fireboss of the order in which to fight multiple fires, and the consequent loss of acreage of forest. A good ordering minimizes the loss of acreage. After watching fires in Phoenix for a long time, one gets a sense of the factors that affect how much area burns, and, thus, the factors that influence the fireboss's decision about the order in which to fight the fires.

Probably Influences Fireboss's Ordering Decision	**Probably Doesn't Influence Decision**	**Unknown or Uncertain Influence on Decision**
Wind speed	Wind direction	Boundaries
Ground cover	Shape of the fire	Nonlinear fire growth
Elevation gradients	Where on the perimeter bulldozers work first	
Direct or indirect attack		
Initial size of fires	Fluctuations in wind speed and direction	
Number of bulldozers		

Table 1. Assessment of factors with respect to the fireboss's ordering decision.

The factors that should most influence the ordering decision are probably wind speed, ground cover, the initial size of the fires, and the force one can bring to bear on the fires (see Table 1). It also matters whether bulldozers work directly at the fire edge (direct attack), or at a distance (indirect attack). The direction of the wind probably does not affect the ordering decision, nor do small fluctuations in wind speed and direction, which cancel out over time. Fires in the Phoenix environment generally have lumpy elliptical shapes, but the exact shape probably has little effect on the ordering decision, and it probably does not matter where bulldozers start working. Natural and artificial boundaries are currently exploited by the Phoenix planner, but it is unclear how these should affect the ordering decision. Another uncertainty is whether the fire perimeter can ever increase at a nonlinear rate that is high enough to affect performance. Preliminary data tell us that perimeter growth is linear, but when convective fires are implemented in Phoenix, they will probably influence the ordering decision (see Sec. 2.3.2).

This assessment leads to some assumptions, and then to a model. I will only model indirect attack (but see Section 2.3.1). I assume fires grow as circles, and that the radius of the circles grow at a constant rate (which can vary from fire to fire but is fixed for a given fire). I also assume that fireline cut by bulldozers is contiguous and its position around the fire is irrelevant. I also assume that the fire grows by the same amount at all points on its perimeter that are not constrained by fireline or other boundaries.

2.2 Modelling

It turns out that if the Phoenix fireboss can minimize the total amount of time that agents require to contain a sequence of fires, it will also minimize the total area burned [3]. The following discussion is a model of how much time is required to contain a series of fires. Section 2.3 derives from this model a rule for determining the order in which to fight the fires.

As I noted in Section 2.1, fire growth is roughly linear. The radius of the fire grows by a constant at each time unit:

$$r(t+1) = r(t) + k$$

Assuming circular fires, the perimeter spreads by $s = 2\pi k$ at each time unit. A fire is usually not noticed immediately by the Phoenix fireboss because its subordinate watchtowers require a significant interval to scan an area. Also, bulldozers need time to reach the fire. So by the time bulldozers begin working on a fire, it already has a significant perimeter, which I denote p_0.

Now, imagine an agent constructs a circular line at a constant rate around a growing fire, as shown in Figure 2, so that when it has finished constructing the line, the fire is completely within the line.

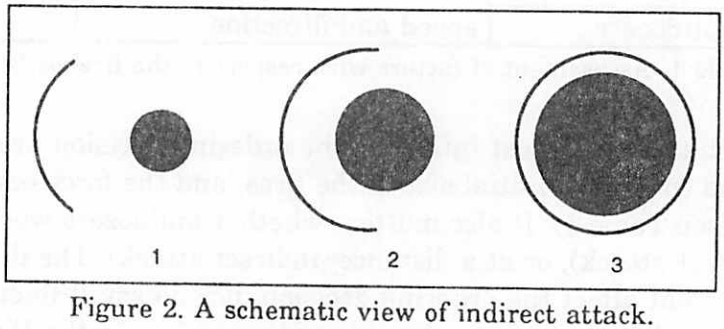

Figure 2. A schematic view of indirect attack.

The time it takes to construct such a line depends on p_0, the initial perimeter of the fire; s, how fast the fire grows; and C, how fast bulldozers can construct fireline. The fire is contained by the line when

$$t \times C = p_0 + t \cdot s \qquad (1)$$

that is, when

$$t = \frac{p0}{C - s} \qquad (2)$$

The situation is as shown in Figure 3, which plots the perimeter of the fire (y axis) against time (x axis). The rate of increase of the fire perimeter is s, compared with a rate of C for line constructed by the agent. As long as C > s, the agent's line will eventually contain the fire. Whether it is sooner or later depends on p_0, C and k, as suggested by Eq. 2. Recall that p_0 is the perimeter of the fire at the time bulldozers start working on it. At some time in the past, the fire was very small, perhaps just a single tree or patch of grass. This point, Q, proves significant later.

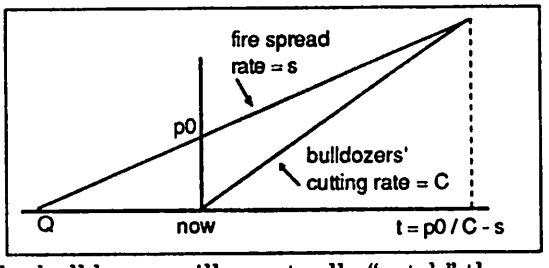

Figure 3. If C > s, the bulldozers will eventually "catch" the growing fire at time t.

Equation 2 can be extended to multiple fires that are fought in succession (and also to fires fought simultaneously, though I will not describe that here). In the Phoenix environment, different fires have different initial perimeters and spreading rates. I denote the initial perimeters of fires $f_1, f_2, \ldots, f_n$ as $p_0(f_1)$, $p_0(f_2)$,...$p_0(f_n)$, the rates of spread as $s(f_1)$, $s(f_2)$,...,$s(f_3)$. If the fireboss attacks fire f_1 before any other fire, then it will require $t(f_1) = p_0(f_1) / C - s(f_1)$ time units to contain (following Eq. 2). By convention, I denote the first fire as f_1, the second as f_2, and so on. However, I will often refer to the time it *would have required* to contain fire f_i if the fireboss had attacked it first. I denote this $t(f_i)$.

To derive the time required to contain a series of fires, consider the time it takes to contain fire f_2 *after* containing fire f_1.

$$t(f_1, f_2) = \frac{(p_0(f_2) + t(f_1)\, s(f_2))}{(C - s(f_2))} \qquad (3)$$

When the buldozers finally start work on fire f_2 (after containing fire f_1), its perimeter will be larger than it would have been if they had worked on it first. $p_0(f_2)$ will have grown at rate $s(f_2)$ for as long as it took to contain fire f_1. Therefore, the perimeter of fire f_2 at time $t(f_1)$ will be $p_o(f_2) + t(f_1)\, s(f_2)$. Expanding Eq. 3 gives

$$t(f_1, f_2) = \frac{p_0(f_2)}{(C - s(f_2))} + \frac{\frac{p_0(f_1)}{C - s(f_1)}\, s(f_2)}{(C - s(f_2))}$$

Now let us introduce a function, $g(x) = s(x) / (C - s(x))$. Rewriting the previous expression in terms of g gives

$$t(f_1, f_2) = \frac{p_0(f_2)}{(C - s(f_2))} + g(f_2)\, \frac{p_0(f_1)}{(C - s(f_1))} = t(f_2) + g(f_2)\, t(f_1) \qquad (4)$$

This is the time to contain fire f_2 after fire f_1. The time to contain *both* fires is

$$T(f_1,f_2) = t(f_1) + t(f_1,f_2) = t(f_2) + (1 + g(f_2))\ t(f_1) \quad (5)$$

We can extend this model to three fires, and then to any number of fires:

$$t(f_1,f_2,f_3) = \frac{(p_0(f_3) + T(f_1,f_2)\ s(f_3))}{(C - s(f_3))} \quad (6)$$

$$\begin{aligned} T(f_1,f_2,f_3) &= T(f_1,f_2) + t(f_1,f_2,f_3) \\ &= T(f_1,f_2) + t(f_3) + g(f_3)\ T(f_1,f_2) \\ &= t(f_3) + (1 + g(f_3))\ T(f_1,f_2) \end{aligned} \quad (7)$$

Eq. 6 is analogous to Eq. 3 and Eq. 7 is analogous to Eq. 4. In general,

$$T(f_1,f_2,\ldots,f_m,f_n) = t(f_n) + (1 + g(f_n))\ T(f_1,f_2,\ldots,f_m) \quad (8)$$

From this analysis, one can see that the order in which one fights fires affects the time required to fight them. Imagine two fires, a and b, with $p_0(a) = 100$, $p_0(b) = 150$, $s(a) = 12$, and $s(b) = 4$; and the maximum rate at which bulldozers can cut fireline is $C = 20$. If we fight fire a first (i.e., assign $\{a \rightarrow f_1, b \rightarrow f_2\}$) then $T(a,b) = 25$. Alternatively, if we fight b first (i.e., assign $\{b \rightarrow f_1, a \rightarrow f_2\}$) then $T(b,a) = 35.9$.

2.3 Design

The Phoenix fireboss should fight fires in the order that will minimize the sum of the times it spends on each. Given a set of fires A = {a,b,...}, we need a rule that maps A onto the ordered set F = $\{f_1,f_2,\ldots\}$ so that $T(f_1,f_2,\ldots)$ is minimized. One approach would be to have the fireboss calculate the time required for each order, then select the order with the lowest time. Because the number of orders increases as the factorial of the number of fires, this approach could be expensive. In fact, there is a much quicker way to find the best order: Each fire f_1 can be characterized by a function of its initial perimeter and its rate of spread:

$$Q(f_i) = \frac{p_0(f_i)}{s(f_i)}.$$

If $Q(a) < Q(b) < \ldots < Q(k)$, and the fireboss fights fires in the order {a,b,...k} then it will minimize the time required to fight the fires. To prove this, I will show that if $Q(a) < Q(b) < , \ldots , < Q(k)$, then the fireboss cannot minimize time unless it fights fire a first. The rule is to fight first the fire with the smallest value of Q. Then, after it has fought the first fire, the fireboss can reapply this rule to the remaining fires. To prove this rule, I will prove two lemmas:

Lemma 1: $T(i,j) < T(j,i)$ iff $Q(i) < Q(j)$

Lemma 2: $T(i,j) < T(j,i) \rightarrow T(S_1, i, j, S_2) < T(S_1, j, i, S_2)$ where S_1 and S_2 are subsequences of zero or more fires.

Lemma 1: From the definitions of Q, t, and g, it follows that $Q(f_i) = t(f_i) / g(f_i)$. I will show that

$$T(i,j) < T(j,i) \quad = \quad \frac{t(i)}{g(i)} < \frac{t(j)}{g(j)} \quad = \quad Q(i) < Q(j)$$

Expanding T(i,j) < T(j,i) according to Eq. 5 gives

$$t(i) + t(j) + g(j)t(i) \quad < \quad t(j) + t(i) + g(i)\ t(j)$$

Eliminating the common terms gives

$$g(j)\ t(i) \quad < \quad g(i)\ t(j) \quad = \quad \frac{t(i)}{g(i)} < \frac{t(j)}{g(j)} \quad = Q(i) < Q(j)$$

So $T(i,j) < T(j,i) \equiv Q(i) < Q(j)$.

Lemma 2: It is intuitively clear that if T(i,j) < T(j,i), then $T(i\ ,j,\ S_2) < T(j,i,\ S_2)$. All fires in the subsequence S_2 are growing while fires i and j are being fought, and they will grow more, and thus take longer to contain, if those fires are fought in the order j,i than if they are fought in the order i,j. Similarly, if $T(S_1,\ i\ ,j) < T(S_1,\ j,i)$ then, by the same reasoning, $T(S_1,\ i\ ,j,\ S_2) < T(S_1,\ j,i,\ S_2)$. It remains to prove that T(i,j) < T(j,i) implies $T(S_1,\ i\ ,j) < T(S_1,\ j,i)$. To begin, we rewrite Eq. 8 with all its terms fully expanded:

$$T(f_1,f_2,\ldots,f_m,f_n) = t(f_n) + (1 + g(f_n))\ (t(f_m) + (1 + g(m))(\ \ldots\ (t(f_2) + g(f_2)(\ t(f_1)\ )\ )\ \ldots\ )\ )$$

Rearranging terms and multiplying through, we get

$$\begin{aligned} T(f_1,f_2,\ldots,f_m,f_n) = \quad & t(f_1)\ (1 + g(f_2))(1 + g(f_3))\ \ldots\ (1 + g(f_m))(1 + g(f_n)) + \\ & t(f_2)\ (1 + g(f_3))\ \ldots\ (1 + g(f_m))(1 + g(f_n)) + \\ & \ldots + \\ & t(f_m)\ (1 + g(f_n)) + \\ & t(f_n) \end{aligned}$$

Note that the last two terms in this sum are $(1 + g(f_n))t(f_m) + t(f_n)$, which is just $T(f_m,f_n)$. If we decide to fight these fires in the opposite order, then the last two terms of the sum become $(1 + g(f_m))t(f_n) + t(f_m) = T(f_n,f_m)$, *but the previous terms in the sum do not change.* If you swap the last two fires, f_m and f_n, in a sequence of fires, the resulting difference in time is just $T(f_m,f_n) - T(f_n,f_m)$. Therefore, the lemma is proved: $T(i,j) < T(j,i) \rightarrow T(S_1,\ i,\ j) < T(S_1,\ j,\ i)$

Now, if we know Q(a) < Q(b) < Q(c) < Q(d), we can use lemma 1 to show

$$\begin{aligned} & T(a,b) < T(b,a),\ T(a,c) < T(c,a),\ T(a,d) < T(d,a), \\ & T(b,c) < T(c,b),\ T(b,d) < T(d,b), \\ & T(c,d) < T(d,c) \end{aligned} \qquad (9)$$

The fireboss should not fight d first: If it does, then it will next have to fight either a, b, or c, giving the sequences T(d,a,...) or T(d,b,...) or T(d,c,...). Because we know from Lemma

2 that if T(i,j) < T(j,i) then T(i,j,S2) < T(j,i,S2), and we know from (9) that T(d,x) > T(x,d) for all x, it follows that for every sequence that begins {d,x,...} there is another with a lower value of T that begins {x,d,...}.

Similar reasoning shows that the fireboss must fight fire a first. If it doesn't, then at some point in the future, it will generate a sequence {S_1, x, a}, and we know from Lemma 2 and (9) that this has a higher value of T than {S_1, a, x}. Now we start "unpacking" the subsequence S_1: for each of its components, T would be smaller if fire a preceded that component. So if Q(a) < Q(b) < Q(c) < Q(d), the time to fight all the fires cannot be minimized unless fire a is fought first. This reasoning applies anew to fire b: if Q(b) < Q(c) < Q(d), then the fireboss must fight fire b before the others. Thus, Q(a) < Q(b) < ... < Q(k) implies that T(a,b,...,k) is the smallest time to fight fires a,b,...,k.

This rule is illustrated graphically in Figure 4. In the top pane, starting at the point labelled "now," the Phoenix agents build fireline around fire a until it is contained. This is shown as the intersection of the heavy "fireline" line and the thinner "fire a" line, at t(a). The agents then start work on fire b (assuming that travel time from fire a to fire b is

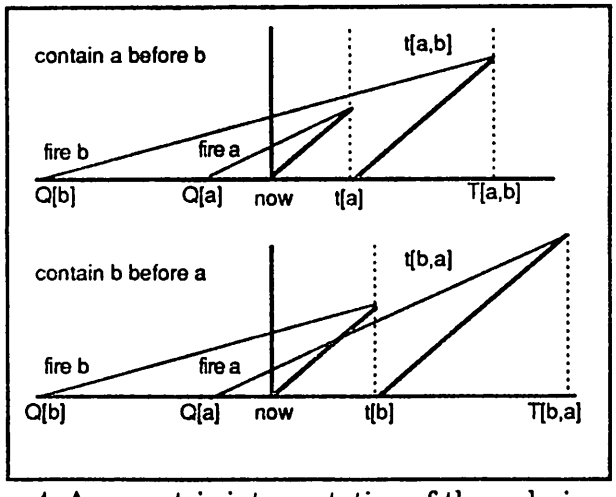

Figure 4. A geometric interpretation of the ordering rule.

negligible). Fire b is contained at time T(a,b). The intervening period between t(a) and T(a,b) is t(a,b). In the bottom pane, this pattern is reversed. Agents work on fire b first. Note that T(a,b) < T(b,a). It was recently discovered by Paul Silvey that Q(a) and Q(b)—which I had treated as decision variables with no particular interpretation—in fact have a natural interpretation as the length of time the fires have been burning. (Silvey noted that since Q(i) = $p_0(f_i)$ / $s(f_i)$, and $p_0(f_i)$ is a "distance" and $s(f_i)$ is a "rate," Q(i) must therefore be a "time," and derived his interpretation from Eq. 1.) The surprising implication of this interpretation is that Phoenix agents should fight the *youngest* fires, not necessarily the smallest, or slowest, first.

Let me summarize the discussion to this point. The premise of the ecological view is that behavior results from the interaction between an agent and its environment, and that we should design agents from models of these interactions. Equations 2 and 8 are models of

this kind because they represent how a measure of behavior (the time to control a sequence of fires) results from the interaction between environmental factors (the rate at which fires spread, s and g), and aspects of an agent's architecture (the rate at which line can be cut, C).[1] From these models, it was possible to design an ordering scheme to miminize time. In closing this discussion, I will briefly list other ongoing modelling efforts:

2.3.1. Direct attack. The current model assumes agents work at a distance from the fire. In fact, Phoenix agents can also work directly at the fire edge. In this case, called direct attack, the perimeter growth is not linear, because as agents control more of the perimeter its growth rate decreases. Empirically, this nonlinearity is quite small, but we need to know whether it can obviate the strategy to order fires by Q.

2.3.2. Nonlinear growth. As fires get bigger they generate convective winds, which increase the rate at which the fires grow. We have not yet implemented convective fires in the Phoenix environment. When we do, we may need to rethink the strategy for ordering fires, because the oldest fires, not the youngest, are the most likely to become convective. Nonlinear growth adds a previously absent dimension to the fireboss's scheduling problem: hard deadlines. In the *linear* model, as long as C > s, delays in fighting a fire have linear effects—the bulldozers will "catch" the fire eventually (Fig. 5.a). But if fire growth is not linear, a delay may ensure that the bulldozers can never contain the fire. When the "C line" is tangent to the "s line" (Fig. 5.b), its x intercept is a hard deadline.

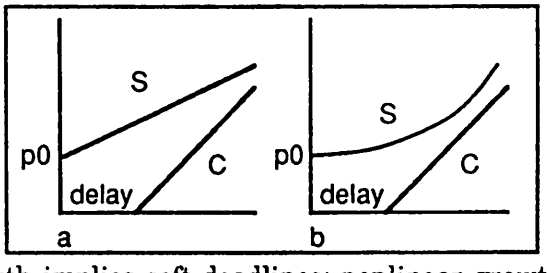

Figure 5. Linear fire growth implies soft deadlines; nonlinear growth implies hard deadlines.

2.3.3. Other performance measures. I have assumed that the fireboss wants to minimize the area burned, but it may want to minimize the total cost of the area burned and the resources it requires to fight the fires. Assuming that the acreage burned increases with the square of time, the cost of a firefighting operation can be modelled as the following sum:

$$\Sigma = (t^2 \cdot \text{cost(acre)}) + (N \cdot \text{cost(bulldozer)})$$

[1] p_0, the size of the fire when the bulldozers reach it, can be treated either as an environmental factor or as another measure of behavior, since it results from the interaction of agent features, such as the time it takes the fireboss to respond to a reported fire, and environmental conditions.

given that N is the number of bulldozers and t is defined by Eq. 2. As N decreases, t increases. We can find the value of N that minizes this sum by setting its first derivative with respect to N to zero and solving for N:

$$N = s + \frac{cost(acre)^{1/3}\, p_0^{2/3}}{.5^{1/3}\, cost(bulldozer)^{1/3}}$$

2.3.4. Other Strategies. Many strategies besides the current one, which allocates bulldozers to fires sequentially, need to be analyzed. I am currently looking at a strategy that divides bulldozers among fires and fights them simultaneously.

2.3.5. Uncertainty. I assumed that the fireboss knows the values of parameters such as p_0 and s, when in fact they are uncertain (indeed, s fluctuates over time). We must extend our models to represent these sources of uncertainty.

3. Prediction, Explanation, and Generalization

The efficacy of designing from models depends on whether models can predict how designs will behave. As AI researchers work with more complex environments, and with architectures that produce complex behaviors from interactions of simpler behaviors, the goal of predicting behavior seems increasingly remote. Some researchers claim that behavior is in principle unpredictable, so the only way to design systems is as Nature does, by mutation and selection (e.g., [9], p. 25). I think this is going too far: We can often predict the behavior of a system at a level of abstraction or aggregation that is useful for design, even if we cannot predict details of the behavior. For example, I do not expect Eq. 2 to predict precisely t, the time it takes to contain a fire. No model can, because t depends on the interactions of hundreds of events over time. But as Figure 6 shows, Eq. 2 does predict the general form of the relationship between t and (C – s): t increases as (C – s) approaches zero. Nonlinearities in performance, like this one, alert designers to diminishing returns: Increasing the number of bulldozers that are sent to a fire (increasing C), will have a increasingly smaller effects on t; whereas decreasing the number of bulldozers will have an increasingly larger effects. I am confident in these trends in the values of t, if not in the values themselves. So the question is not whether predicting behavior is possible in principle, but whether prediction is useful in practice.

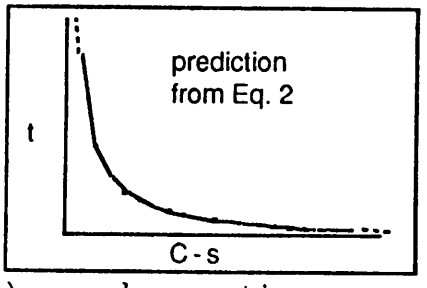

Figure 6. As (C – s) approaches zero, t increases as a harmonic series

Quite apart from their utility to design, modelling and prediction are useful for analysis of systems that already exist—for explaining why systems behave as they do. And here, predictions can be equally useful if they are wrong, because they prompt revisions in our models. For example, we recently noticed that Phoenix's predictions about the fire spread were wrong, adversely affecting its performance. The problem was that fires spread much more quickly upwind than predicted. Scott Anderson discovered that the fire behaved in a way that none of us anticipated or observed: it tacks upwind, moving from point A to point B by tacking first from A to C, then from C to B; and it does this much more quickly than going from A to B directly. Anderson has now revised the fire-spread model.

But, returning to design for a moment, how can we design from models that sometimes make wrong predictions? I acknowledge this concern, but what is the alternative? Currently, we design AI programs to vague specifications; for example, we designed Phoenix agents for "unpredictable, real-time, dynamic" environments, and endowed them with "reactivity, approximate processing, responsiveness," and so on. With such vague design goals, it is hard to know whether we succeeded. Phoenix agents put out fires, of course; but as Adele Howe and I have discussed, demonstrations are not adequate evaluations because they don't tell us *why* a system works or doesn't work [2,5]. And even good evaluations cannot ensure that our results are general, because you and I may interpret the informal terms we use as design goals, such as "real time," differently.

4. Summary

Modelling might provide a solution to these problems. A model is a formal summary of our understanding of how behavior emerges from the interactions between an agent and its environment. Models are no less useful for being formal. I think we are better off designing from models like Eqs. 2 and 8 than we are designing from informal terms like "real time." And if a model is inaccurate, if it makes wrong predictions, that is because our understanding of the environment is wrong; it has nothing to do with whether we express that understanding formally or informally. On the other hand, formal models like Eqs. 2 and 8 make our design goals precise. This means that we can say exactly what a system does, instead of relying on demonstrations that a system "works." This is essential to achieving general results in AI. For example, Figure 5 shows a sketch of one definition of "hard deadline": the tolerable delay before a linear process starts "chasing" a superlinear one. You may have another definition. If our definitions are precise, and encapsulated in models, then we can reason about our respective models, instead of guessing what our respective programs might or might not do.

4. References

1. Chapman, D. and P. E. Agre. Abstract Reasoning as Emergent from Concrete Activity. Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon. 411-424, 1987.

2. Cohen, P. R., Howe, Adele E. "How evaluation guides AI research." AI Magazine. 9(4): 35 - 43, 1988.

3. Cohen, P. R. Discovering Functional Relationships that Model AI Programs. 1989.

4. Cohen, P. R., Greenberg, M. L., Hart, D.M., Howe, A. E. "Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments." AI Magazine. 10(3): 32-48, 1989.

5. Cohen, P. R. and A. E. Howe. "Toward AI research methodology: Three case studies in evaluation." IEEE Transactions on Systems, Man and Cybernetics. 19(3): 634-646, 1988.

6. Cohen, P. R., A. E. Howe and D. M. Hart. Intelligent Real-time Problem Solving: Issues and Examples. Intelligent Real-Time Problem Solving (IRTPS). 1989.

7. Dean, T. L. Intractability and Time-dependent Planning. Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon. 245-265, 1987.

8. Kaelbling, L. P. An Architecture for Reactive Systems. Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon. 395-410, 1987.

9. Langton, C. Artificial Life. Santa Fe Institute Studies in the Sciences of Complexity. 1989.

10. Neisser, U. "Cognition and Reality." 1976 Freeman Press. San Francisco.

11. Rosenschein, S. J., B. Hayes-Roth and L. D. Erman. Notes on Methodologies for Evaluating IRTPS Systems. Intelligent Real-Time Problem Solving (IRTPS). 1989.

12. Simon, H. A. "The Sciences of the Artificial." 1981 The MIT Press. Cambridge, MA.