Envelopes as a Vehicle for Improving the Efficiency of Plan Execution

David M. Hart, Scott D. Anderson, Paul R. Cohen

COINS Technical Report 90-21

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

Abstract

Envelopes are structures which capture expectations of the progress of a plan. By comparing expected progress with actual progress, envelopes can notify the planner when the plan violates those expectations. The planner then has the opportunity to modify the plan to increase its efficiency given the unexpected progress. This paper presents a specific example of the construction and use of an envelope, followed by a discussion of the general utility of envelopes for improving the efficiency of plan execution.

This research was sponsored by DARPA-AFOSR contract F49620-89-C-00113; the Office of Naval Research, under a University Research Initiative grant, ONR N00014-86-K-0764; the Office of Naval Research, contract # N00014-88-K-0009; and a grant from the Digital Equipment Corporation.

1 Introduction

Most AI planners test the postconditions of an action after its completion to see if it succeeded, but in our domain, actions take so long to execute that advance knowledge of the probable outcome is valuable. Therefore, we monitor actions *during* execution. We represent the *a priori* expectations of action progress, which we compare with the actual state, in structures we call *envelopes*. By comparing actual progress to the expectations about progress stored in envelopes, we can see whether a plan[1] is executing better or worse than we expected.

Inefficiency in plan execution encompasses wasteful use of resources in a plan that is succeeding, ineffective use of resources in failing plans, and even the costs of recovering from a failed plan. Envelopes are chiefly concerned with the efficiency of plan execution, and only indirectly with the planning process. In this paper, we will show an envelope that categorizes the progress of a plan as *better*, *worse*, or *as* expected. If progress is better than expected, we can increase the efficiency by reducing resource expenditure; if worse, we can act to avert plan failure, possibly by adding resources.

The term "envelope" derives from the idea of a "performance envelope" in engineering, describing the performance profile of a mechanism under various conditions. When the performance of a plan in our system goes outside its envelope, the plan may no longer be appropriate for the current environmental conditions. This paper details the construction and use of a particular envelope in a multiagent, real-time problem solving system that fights simulated forest fires. It also discusses the general utility of envelopes for improving the efficiency of plan execution.

2 Monitoring Plan Execution

There is an obvious advantage to knowing how a plan is progressing when the planner can act based on that knowledge; if the plan is failing, the planner can either abort the plan (avoiding throwing good resources after bad) or add resources to the plan so as to avert the failure. In domains where actions are

[1]Because our plans are structures of one or more actions, we will use the terms interchangeably.

not interruptible or are of such short duration that there is no time to add or subtract resources from an action in progress, there is clearly no utility to monitoring an executing action. But fighting a forest fire can require the efforts of many agents over many days, so that plans execute over long time spans and there is ample time to add and subtract resources. Therefore, in our domain (simulated in a testbed called Phoenix [1]), it pays to monitor a plan during its execution.

Doyle's work [2] addresses monitoring, but in a robotics domain using a STRIPS-style action model with specified preconditions and postconditions. Monitoring verifies the truth of the preconditions and postconditions:

> ...we assume that the successful execution of actions can be verified by instantaneously verifying the action's preconditions before its execution and instantaneously verifying its postconditions after its execution. This approach proves inadequate for some actions. [2]

Doyle goes on to describe cases that require just the sort of continuous monitoring that envelopes are designed for, such as actions that are extended over time and can fail at any point, and actions involving looping, which can be viewed as extended action.[2]

Our work is closest in spirit to that of Sanborn and Hendler [6]. Their simulated robot, which tries to cross a busy street, must monitor the objects in the world (cars rushing past) and predict whether they will run over it. We view this as an envelope around the plan of crossing the street, attempting to avert the catastrophic failure of the plan, not to mention the robot. This is a clear case of an extended action (though performed as a loop of single steps forward and backward) in which forewarning of failure is critical. The forewarning is achieved by predicting the location of the cars and the robot; we will see the significance of prediction for envelopes below.

Envelopes have been implemented as a general mechanism in Phoenix. Performance falling outside expected bounds is termed a "violation," and violations notify the planner so that its planning knowledge can be brought to bear on the situation. The purpose of envelopes, then, is to provide information to the planner that guides its decision-making during plan excution.

[2]The example of looping that he gives is of filling a bucket from a hose, which we believe is more naturally viewed as an extended action.

While the planner has a number of options, it typically responds to violations as we have mentioned—adding or subtracting resources. Without envelopes, these opportunities to increase efficiency would go unnoticed.

3 Constructing an Envelope

In Phoenix, simulated forest fires are controlled by bulldozers cutting fireline around the fire. In some cases, it is too dangerous for bulldozers to cut a fireline close to the fire, and so we use what is called "indirect attack," in which the bulldozers cut a line some distance away. In indirect attack, a central fireboss coordinates the actions of the bulldozers. For example, in Figure 1, the intended placement of fireline, to be cut by several bulldozers, is the polygonal shape surrounding the fire. The fireboss selects a polygon such that the estimated time required for n bulldozers to cut the line, $BT(n)$, is less than the estimated time remaining until the fire spreads to the polygon, FST; the difference is the amount of slack time in the plan.

Figure 2 illustrates how an envelope for the multiple-bulldozer, indirect-attack plan is constructed. We define "progress space" as the percentage of the fireline which is completed, PFC, versus time (elapsed simulation time, t). The point at the upper right is the estimated time that the fire arrives at the polygon, t_{fa}, and 100% of the fireline is dug. Lines 1 and 2 are defined by the expected rate that some number of bulldozers can cut fireline: line 1 has a slope of $100/BT(n)$, because n bulldozers must cut 100% of the line; line 2 has a slope of $100/BT(n-1)$. t_p is the time that these estimates were made and the envelope was built. t_{ls} is the latest time at which n bulldozers can start digging line and expect to finish the fireline before the fire arrives.

The filled circle labeled CP represents the current position in progress space—(t_{now}, PFC_{now}). The location of CP within the regions of progress space indicates how the plan is progressing: crossing below line 1 suggests that the plan will fail, since the bulldozers would need to cut the fireline at a rate faster than we think they can;[3] crossing above line 2 suggests that we should save resources by retiring a bulldozer, since $n-1$ of them should

[3]The plan will not *necessarily* fail, since the bulldozers might do better than we expect, even though they so far have not, or the fire might take longer to reach the fireline than we thought, say if it rains.

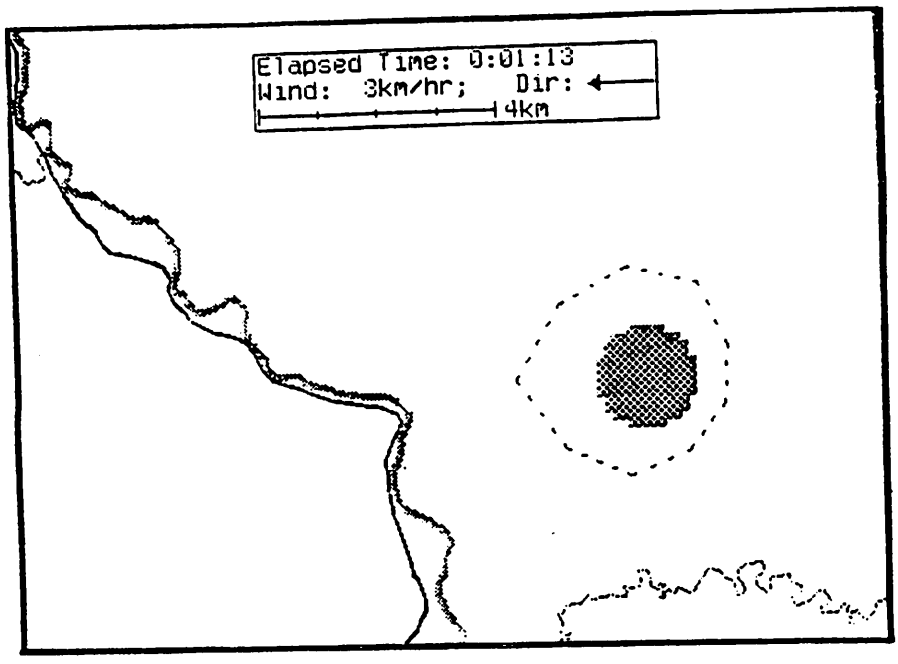

Figure 1: A fire (shaded region) has been set in the Phoenix forest, seen here 1 hour 13 minutes afterwards. The wind is from the East at 3 kph. The polygon of dashed lines marks both the projected shape of the fire after about 17 hours and the intended placement of fireline for the indirect-attack plan. Other lines are rivers and roads.

be able to cut the remaining line in time. The area between the lines is the envelope—the range of expected performance—and going outside it is a violation of the envelope, signalling to the agent that something should be done.[4]

4 Using Envelopes

The plan library of each Phoenix agent contains skeletal forms of envelopes as well as plans. Part of the definition of a plan denotes what

[4]Conceptually, the envelope is just that area of progress space, but we also use the term to describe the data structure representing this area and associated code for creating the envelope and updating CP.

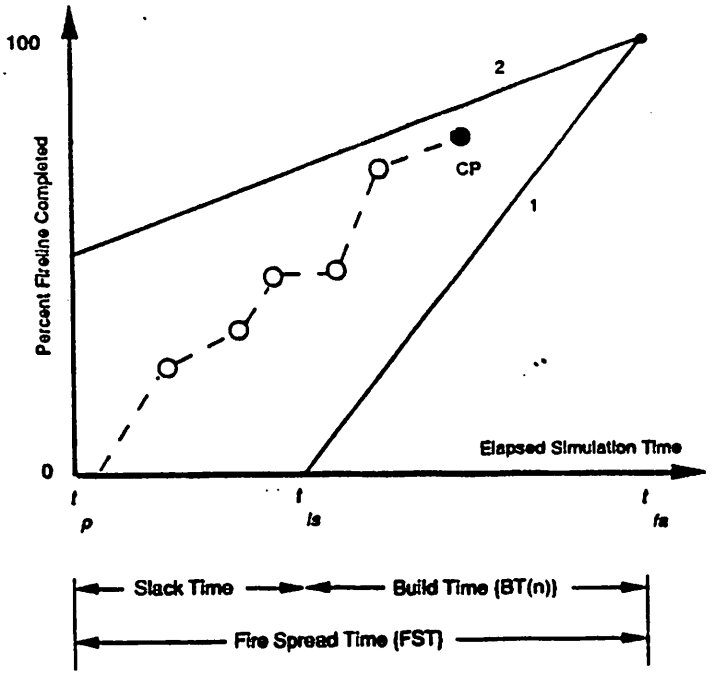

Figure 2: An envelope for the Indirect-Attack Plan

envelopes, if any, should be instantiated to monitor the execution of that plan. Consequently, when a plan is instantiated at run-time, the associated envelopes are also instantiated, and the envelopes initialize themselves from plan variables, such as the $BT(n)$ and t_{fa} variables above. Each envelope provides a *monitor method* whose purpose is to update the current progress (CP) and locate it in progress space. Then, while the plan is executing, a periodic action called "monitor-envelopes" causes the agent to verify that the plan is progressing satisfactorily by running these monitor methods.

The monitor method checks sensory information previously gathered and stored in the plan variables, such as the current positions of the bulldozers and the fire, and determines which region CP lies in. If CP is within the region for acceptable progress, nothing more happens. However, if CP has crossed into a region of unexpected progress (either better or worse), the envelope is violated and the monitor method adds an item to the agent's agenda so that the agent can notice and respond to the violation. In Figure 3, we show an envelope in which the CP falls into the failure region.

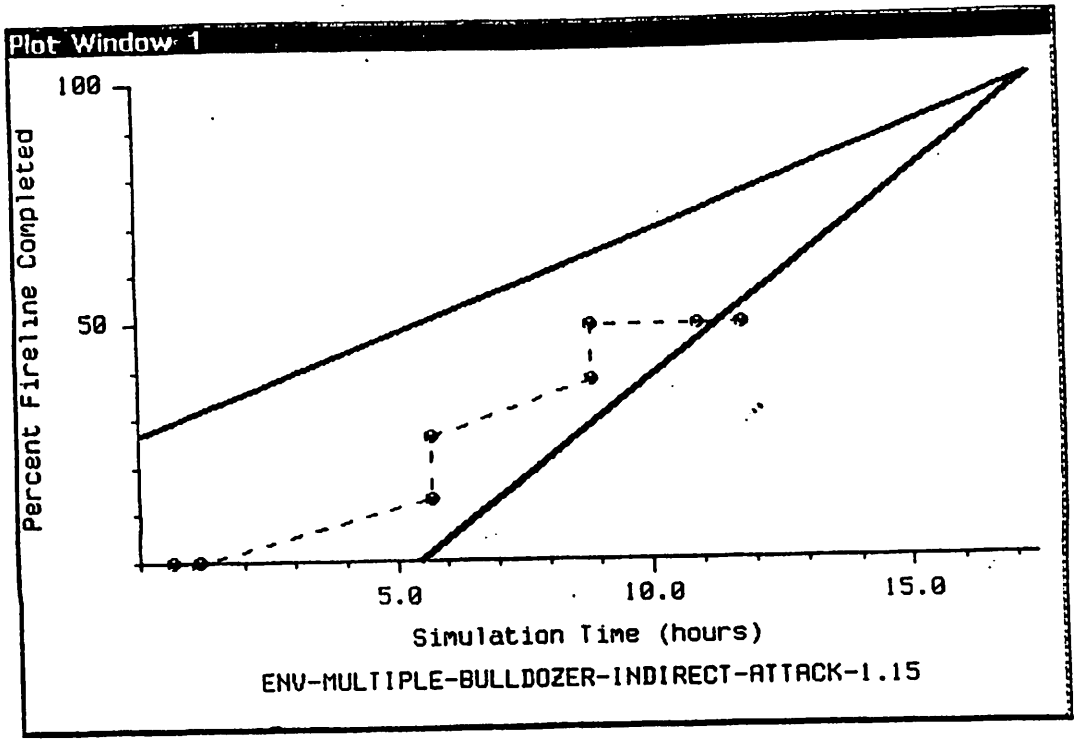

Figure 3: Plot of the Multiple-Bulldozer, Indirect-Attack Envelope in Phoenix, showing a violation at nearly 12 hours into the simulation. The recent progress is flat because the bulldozers have stopped to refuel.

While the fireboss could do many things as a result of this violation (for example, the fireboss might buy time, say by dumping fire retardant on the fire or expanding the polygon around the fire), consider the case in which it sends another bulldozer to help dig line. A new envelope must be set up for monitoring this modified plan. The failure boundary for the new envelope will be determined by a line whose slope is $100/BTn+1$. The additional bulldozer is sent, and the fire is successfully contained within the original polygon, which means the modified plan has succeeded, where the original plan would almost certainly have failed.

We have mentioned that multiple envelopes might exist simultaneously; one way that this occurs is when a plan and a step in the plan both have envelopes. For example, in the multiple-bulldozer, indirect-attack plan, each step of digging a side of the polygon has an envelope. This allows the fireboss to apportion the time constraints from the overall plan to the steps in the

plan. A violation of a step's envelope may indicate a problem with the whole plan, or may simply mean that other steps will have to do better than expected. Therefore, the step envelopes do not eliminate the need for the plan envelope—the latter integrates the information from step envelopes.

Furthermore, since digging a side of the polygon will in fact be executed by bulldozers and not by the fireboss, the envelope for that step must be an explicit data structure that can be communicated to the bulldozers. We call this an *agent envelope*, since it is monitored by the agent who receives it, allowing the fireboss to turn its attention elsewhere. If a violation occurs, the agent reports back to the fireboss, who assesses the significance of the violation by consulting the plan envelope. Agent envelopes free the fireboss from the task of monitoring the progress of component plan steps carried out by other agents—the fireboss assumes the agent's progress is within expectations unless it receives a violation report from the agent. Powell and Cohen [5] discuss the use of envelopes to coordinate activities among echelons in multiagent, operational planning.

Another way that multiple envelopes occur is when data dependencies exist between envelopes. For example, the estimate of when the fire will reach the polygon, t_{fa}, is crucial to the way that progress space is carved into regions. Unfortunately, that estimate is based on inexact information, because the fireboss does not know exactly where the fire is or exactly what the local wind conditions are. Therefore, we put an envelope around our estimate and periodically re-calculate it. Should the re-calculation indicate that the t_{fa} estimate is quite wrong, the t_{fa} envelope is violated, and this causes the envelope on the plan to be revised.

5 Utility of Envelopes

It is intuitively obvious that information about the progress of a plan cannot hurt and could well prove invaluable. However, the agent must put effort into gathering this information, and therefore the cost of envelopes must be worth their benefit. To summarize the possible benefits, using envelopes allows an agent to:

- modify a failing plan so as to prevent its failure
- abandoning a plan that is irretrievably failing.

- retire surplus resources from a plan that is going unexpectedly well.
- improve a plan that is going unexpectedly well. For example, in the multiple-bulldozer, indirect-attack plan, it can move the vertices nearer the fire so as to reduce the forest lost.
- reduce communication overhead between cooperating agents via agent envelopes, that is, by allowing them to share expectations and only communicate when those expectations are violated.

Envelopes cost the planner in three different ways: the cost of setting them up, the cost of monitoring them at time intervals, and the cost of responding to violations. Assessing these costs is complicated by the way they can be traded off against each other. For example, a planner could create a "quick and dirty" envelope, using estimates that are of low quality but quick to compute. Employing such an envelope burdens the planner later, since spurious violations are more likely and time spent responding to them will be wasted. On the other hand, a planner can put a lot of time and effort into creating a great envelope, with precise boundaries based on the best information, resulting in regions that categorize the situation quite well. Violations of a high-quality envelope can be better trusted to indicate a problem with the plan. We can create such high-quality envelopes in Phoenix for some activities, but the cost of creating them is high. For example, the speeds with which a bulldozer travels and digs line can be predicted quite accurately using expensive operations that iterate over the points on its route and sum the costs. The benefit is that the envelope reflects very closely the probable time required to dig a segment of fireline. The cost is the expense of calculating this information, a cognitive task that competes with other necessary cognitive activities for available "thinking" time[5]. This tradeoff balances the cost of building an envelope now with the costs of responding to violations later.

Another tradeoff is in the monitor methods of the envelopes. It's important not to make these too time-intensive, since the cost will be incurred many times over the course of plan execution. For instance, how old should sensory information be before the monitor method discards it and measures the environment anew? One option is to use quick, inexact procedures in

[5]For more on Phoenix agents' cognitive structure, see [1]

the monitor method and then, if a violation occurs, double-check them with better procedures to verify whether the violation is spurious.

In very time-pressured situations, an agent will probably want to choose quick and dirty ways of doing things (including building, monitoring and responding to envelopes), while in less pressured situations, it will probably want to choose higher quality methods; therefore, the Phoenix agent architecture allows for this choice.

An extreme tradeoff is to eliminate envelopes entirely and deal with the plan failure when it arises, that is, dispense with monitoring the progress of the executing plan and rely on reports from the field that the plan has failed. Clearly, in Phoenix, we will have to repair or replace the current plan when the fire is reported to be escaping from the incomplete polygon. The cost of this failure, besides the time to replan, is the loss of more forest and the additional time and effort of bulldozers to control it. The cost of failure must be compared to the costs of using envelopes, which we've noted will depend on the choices made by the agents. We believe that on average these costs will exceed the overhead cost of using envelopes. The same argument can be made with respect to improving a plan that is succeeding: if we can save a little forest by moving the vertices towards the fire when the polygon is being dug faster than anticipated, does this outweigh the cost of using envelopes?

These tradeoffs imply that envelopes have limited utility for some environments and tasks. If the environment is highly variable, so that the estimates and predictions that are built into the envelope don't last, and so that any envelope will be violated shortly, the overhead costs may swamp any benefits. On deeper reflection, though, since envelopes are used for actions that assume some constancy to the environment, such actions would not be used in these highly variable domains. Predictability is the key issue: prediction, used either for planning or building envelopes, is simply not useful in these unpredictable domains. As we mentioned in Section 2, Sanborn and Hendler's simulated robot depends on the predictability of the cars' paths, even though the domain is highly variable. Our approach differs from theirs because they have tightly connected the predictions to the robot's actions, while we notice a violation and let the planner deliberate as long as it wishes over how to solve the problem. We do this because actions can be quite costly; for example, sending another bulldozer to the fire is time-consuming

and commits resources which might be required elsewhere.

6 Current and Future Work

We have integrated envelopes into the Phoenix testbed, and added instrumentation to assess the cognitive load on the various agents. We will test the utility of envelopes by comparing the performance of agents with and without them. In particular, we will run a number of different fires in the simulator, varying the factors contributing to unexpected success or failure of plans, such as weather, mechanical breakdowns, and obstacles. Performance will be assessed based on a combination of cost factors such as forest burned, bulldozer time spent, agents lost, and cognitive overhead incurred. We also intend to experiment with the tradeoffs mentioned above, such as balancing the cost of setting up an envelope with the cost of responding to violations.

Another line of research views envelope violations as an opportunity for learning [3]. Envelopes provide information about plans vulnerable to failure, as well as an opportunity to test ways of repairing plans. These repairs, when successful, can be used to modify the plan library.

We also intend to apply envelopes to the problem of assessing the progress of tasks that involve, not acting in the world, but thinking. Many thinking tasks in Phoenix, such as path planning and predicting fire spread, can be computationally expensive and the time available for them is limited. If we can model these computational tasks so that we can predict their progress well enough to use envelopes, we can control them and increase their efficiency. A brief discussion of the use of envelopes for real-time control appears in [4].

7 Conclusion

We have shown an example of how to build and use an envelope for a plan in a fire-fighting domain. This envelope notices when the plan is failing or succeeding too well. The planner can then adjust the plan to the changing conditions, thereby increasing the efficiency of its execution by minimizing the loss of forest and other costs. We've argued that the predictability of the environment indicates whether envelopes will be useful: if the domain is too

predictable, plans cannot fail, so there is no point to monitoring them, and if the domain is too unpredictable, plans would not be of a duration for which envelopes would be useful. For the middle ground—environments that are uncertain but not too unpredictable, which we think are quite common—we argue that the benefits derived from the opportunity to increase plan efficiency will outweigh the costs of creating, monitoring and responding to envelopes.

Acknowledgements

The authors wish to thank Adele Howe, Dorothy Mammen, Jerry Powell, and Paul Silvey for helpful comments on drafts of this paper and also Mike Greenberg and David Westbrook for their skilled programming support.

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-89-C-00113. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

[1] Paul R. Cohen, Michael Greenberg, David M. Hart, and A.E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, Fall 1989.

[2] Richard J. Doyle, David J. Atkinson, and Rajkumar S. Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. American Association for Artificial Intelligence.

[3] A.E. Howe. Integrating adaptation with planning to improve behavior in unpredictable environments. In *Proceedings of AAAI Spring Symposium on Planning in Uncertain, Unpredictable or Changing Environments*, Palo Alto, CA, March 1990.

[4] A.E. Howe, David M. Hart, and Paul R. Cohen. Addressing real-time constraints in the design of autonomous agents. To appear in *Real-Time Systems*, 1990.

[5] Gerald M. Powell and Paul R. Cohen. Operational planning and monitoring with envelopes. In *Proceedings of the IEEE Fifth AI Systems in Government Conference*, Washington, DC, 1990.

[6] James C. Sanborn and James A. Hendler. Dynamic reaction: Controlling behavior in dynamic domains. *International Journal of Artificial Intelligence in Engineering*, 3(2), April 1988.