IN-63-TM

91523

p-8

The Entropy Reduction Engine: Integrating Planning, Scheduling, and Control

MARK DRUMMOND
JOHN BRESINA
SMADAR KEDAR

AI RESEARCH BRANCH, MAIL STOP 244-17
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035

(NASA-TM-107867) THE ENTROPY REDUCTION ENGINE: INTEGRATING PLANNING, SCHEDULING, AND CONTROL (NASA) 8 p

N92-25741

Unclas

G3/63 0091523

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-91-17

February 1991

The Entropy Reduction Engine: Integrating Planning, Scheduling, and Control

Mark Drummond, John Bresina, and **Smadar Kedar**

Sterling Federal Systems
AI Research Branch, Mail Stop: 244-17
NASA Ames Research Center
Moffett Field, CA 94035 U.S.A.

February 1991

Abstract

This paper describes the Entropy Reduction Engine, an architecture for the integration of planning, scheduling, and control. The architecture is motivated, presented, and analyzed in terms of its different components; namely, problem reduction, temporal projection, and situated control rule execution. Experience with this architecture has motivated the recent integration of learning, and this paper also describes the learning methods and their impact on architecture performance.

This paper appears in
The Proceedings of the 1991 AAAI Spring Symposium
in the session on Integrated Intelligent Architectures
(Stanford University, March 27–29, 1991)

The Entropy Reduction Engine: Integrating Planning, Scheduling, and Control*

Mark Drummond John Bresina† Smadar Kedar
Sterling Federal Systems
AI Research Branch, NASA Ames Research Center
Mail Stop: 244-17, Moffett Field, CA 94035

Abstract

This paper describes the Entropy Reduction Engine, an architecture for the integration of planning, scheduling, and control. The architecture is motivated, presented, and analyzed in terms of its different components; namely, problem reduction, temporal projection, and situated control rule execution. Experience with this architecture has motivated the recent integration of learning, and this paper also describes the learning methods and their impact on architecture performance.

1 Introduction

This paper describes the Entropy Reduction Engine, an architecture for the integration of planning, scheduling, and control. The next section motivates the architecture, and the main body of the paper describes the architecture according to various dimensions. This architecture has been tested on some simple but representative problems, and unless otherwise noted, all capabilities that are described have been implemented.

2 Architecture Motivation

At the outset, we would like to indicate what we mean by the word "integrated", as used in the phrase "integrated architecture". To do this, we must first discuss our target problem class.

We are addressing a class of problems that heretofore have been largely considered in piecemeal fashion. The problems are those that require planning, scheduling, and control. The Entropy Reduction Engine (ERE) project is a focus for research on planning and scheduling in the context of closed-loop plan execution. The objective of the project is to create a set of software tools for designing and deploying integrated planning and scheduling systems that are able to effectively control their environments (Bresina & Drummond, 1990). This objective has two important subgoals: first, we are working to integrate *planning* and *scheduling*; second, we are studying plan execution as a problem in *control*.

Traditional AI planning is mainly concerned with the *selection* of actions that are relevant to achieving given goals. Various disciplines, principally Operations Research, and more recently AI, have been concerned with the *scheduling* of actions; that is, with sequencing a given set of actions in terms of metric time and metric resource constraints. Unfortunately, these two bodies of work remain theoretically and practically disconnected from each other. It is clear that the choices made in planning must influence subsequent scheduling, but it is also true that choices made in scheduling can engender further planning activity. The ERE architecture integrates planning and scheduling functions so that scheduling decisions can give rise to further planning activity.

Most planning and scheduling work assumes that the job of the software system is done when a plan has been generated. However, as Dwight D. Eisenhower observed, "Plans are nothing, planning is everything". We agree with this view in the sense that the importance of planning does not lie in the existence of a *single* plan, but rather in a system's ability to predictively manage plan execution in light of continuous feedback from an environment and to re-plan when failures occur. In the ERE project, we formalize plan execution as a form of closed-loop control, where a plan describes a desired behavior, and feedback from the environment is used to measure deviations.

Diversity in the class of problems poses both the difficulties and opportunities of architectural integration. We are using various problem-solving methods in our architecture, including problem reduction, temporal projection, and rule-based execution. We also intend to employ both analytic and inductive learning methods. The problem-solving methods that are being integrated have been selected due to their apparent relevance to problem types of concern (planning, scheduling, and control). We are pursuing this integration effort with the hope of achieving more from the methods' interaction than could be achieved from any single method in isolation.

3 Architecture Components

The ERE architecture consists of three major components: the Reductor, the Projector, and the Reactor. The Reductor synthesizes appropriate problem-solving strategies for a given problem; the Projector uses these strategies as search control to plan and schedule appropriate actions; and the Reactor executes control rules derived from the Projector's plans. Bresina and Drummond (1990) present an overview of this architecture; here, we only discuss how ERE can be used to build a system for a particular problem. We take a programming language view on the architecture and consider the different sorts of knowledge that a user must provide in order to construct an application system.

First, a user *must* provide a *causal theory* for the domain of application, or no behaviors can be produced by the system. This theory consists of a description of the control *actions* that can be taken by the system, their preconditions, and probability distributions of these actions' possible effects. In a similar fashion, a user can also specify exogenous *events* that are outside of the system's control. This information is used by the Projector to reason about possible system behaviors. To complete the causal theory, the user must provide *domain constraints* that specify those facts which can never co-occur; these constraints are used throughout the system to

*We would like to acknowledge co-funding for this activity provided by the DARPA Information Sciences Technology Office under DARPA Order 7382.

†Also affiliated with the Computer Science Department at Rutgers University.

By *taskability* we mean the ability of a system to accept new goals at run time. ERE has this ability, provided that there is *time* for the three components to react. The Reactor can accommodate a new goal with ease, provided that it has appropriate SCRs. If no such SCRs exist, then the Projector must carry out a search and compile SCRs for the new goal. If the current strategy is inappropriate for the new goal, then the Reductor must respond by producing a new and appropriate strategy. It is possible to change goals at run time because we do not wire a single goal into the system at design time. Of course, there is a cost associated with this run-time flexibility: in order to produce a new set of SCRs, each system component might have to carry out extensive computation.

5.4 Adaptability

By system adaptability we refer to the introduction of learning methods that enable the system to improve itself over time. One way to accomplish this is by acquiring and refining various sources of knowledge. Currently, there are three types of knowledge in the system that are acquired and refined: the causal theory, the SCRs, and the problem reduction rules. In this section, we consider each in turn.

First, the causal theory can be refined by detecting and recovering from failures (Kedar, Bresina, & Dent, 1991). In particular, errors in the causal theory are detected when there is a discrepency between what was projected to occur and what actually occurred during reaction. These 'prediction failures' may be caused by one (or more) of the following: missing operator preconditions, missing operator outcomes, or missing domain constraints. We have implemented a technique using explanation-based learning (Mostow & Bhatnagar, 1987; Gupta, 1987; Chien, 1989; Minton, 1988b) to acquire general preconditions to be added to an operator schema. We are currently developing techniques to handle other kinds of missing information. These techniques require the addition of induction to explanation-based learning.

Secondly, SCRs can be acquired and refined through caching goal-satisfying behaviors synthesized by the Projector. The most specific version of this process is a form of knowledge compilation. However, SCRs can be formed with varying degrees of generality using a goal regression algorithm (Mitchell, *et al.*, 1986) that we have extended to regress goals with temporal extent. Compiling general SCRs using goal regression is very similar to Soar's chunking mechanism (Laird & Rosenbloom, 1990) and to Theo's use of explanation-based learning (Mitchell, 1990).

Thirdly, problem reduction rules can be refined in the face of inappropriate problem-solving behavior. Recall that the Reductor synthesizes an ordered set of subproblems which the Projector attempts to satisfy in turn. Whenever a subproblem is solved, the solution is compiled into SCRs which are immediately made available to the Reactor. Thus, if execution must begin before a complete solution has been found, the Reactor is guided by the SCRs resulting from the current solution prefix. However, harmful interactions between subproblems can make it impossible to extend the partial solution to satisfy the entire strategy. If this happens the Reactor may have to physically backtrack. Hence, it is important that the probability of backtracking over a subproblem solution be kept low. This probability can be reduced by incrementally refining the problem reduction rules, such that the subproblems they specify are more independently solvable. We are currently developing a combination of analytic and inductive learning mechanisms to address this issue.

5.5 Scalability

Any given problem can be scaled up in a number of ways; in general, the major impact on our architecture is an increase in the size of the projection search space. In order for the synthesized SCRs to be of sufficient "quality" within the available time, the search guidance supplied by the Reductor must also be of sufficient "quality". The quality of the synthesized problem solving strategies is dependent (mainly) on the expertise encoded in the problem reduction rules. Thus, if the expertise appropriate for the scaled-up problem is supplied (or learned), the Reductor has the potential to effectively guide the Projector's search. However, this potential can only be realized if there is enough computation time available. Scaled-up problems tend to require deeper searches in the reduction space and in the projection space; hence, the anytime characteristics will degrade. Partially due to this degradation, if all planning search must be carried out concurrent with action, the Reactor's response time could increase. The other (potential) reason for increased response time is that scaled-up problems tend to require a larger set of SCRs to specify the appropriate control advice. We're currently studying system performance in the TileWorld simulator (Philips & Bresina, 1991), but plan to address the scalability issue by using ERE on a more realistic NASA domain in the future.

5.6 Reactivity

Environmental change potentially affects all three ERE components. The Reactor can easily respond to environmental change, and in fact, that is exactly its job. On each cycle, it checks for applicable SCRs (relevant to the given goal and enabled in the current situation) and executes one of the indicated actions. Clearly, if the sensor values indicate a situation for which there is an applicable SCR, then the Reactor can accommodate environmental change. Things are more problematic when we consider how the Projector can respond to environmental change. This problem is essentially that of having a planning system recognize when assumptions it has made about the environment are invalidated during plan construction. We do not currently have a solution to this problem implemented, but we are considering various dependency tracking mechanisms. Lastly, the Reductor must also notice environmental change, since it is possible that the currently selected strategy is inappropriate in the current situation. This is an open research problem.

5.7 Efficiency

The maximum response time of the Reactor occurs when it is *forced* to respond (*i.e.*, perform some action) and no available SCR is applicable. Since the causal theory specifies all possible actions, the Reactor can respond by (randomly) selecting one of the enabled actions. Hence, the maximum response time equals the time needed to determine that no SCRs are applicable plus the time to determine which actions, specified in the causal theory, are enabled in the current situation. This calculation depends on the match algorithm, as well as on the size and, more importantly, on the organization of the set of SCRs. The time required to guarantee an "appropriate" response depends on whether the necessary planning has been done in advance, and if not, it depends on the anytime characteristics of the Reductor and Projector. We have not yet addressed the utility problem; that is, the problem of SCRs with excessively expensive applicability conditions. This problem is especially acute when using generalized SCRs (Minton, 1988a).

Brooks, R.A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1.

Chien, S.A. 1989. Using and Refining Simplifications: Explanation-Based Learning of Plans in Intractable Domains. *Proc. of the Eleventh International Joint Conference on Artificial Intelligence* Morgan Kaufmann, Los Altos, CA.

Christiansen, A., Mason, M., and Mitchell, T. 1990. Learning Reliable Manipulation Strategies Without Initial Physical Models. In *Proc. of the IEEE International Conference on Robotics and Automation.*

Currie, K. and Tate, A. 1985. O-Plan: Control in the Open Planning Architecture. In *Proc. of the British Computer Society Expert Systems '85*, Warwick, U.K., pp 225–240. Cambridge University Press.

Dean, T., and Kanazawa, K. 1989. A Model for Projection and Action. *Proc. of IJCAI-89.* pp. 985–990.

Drummond, M., and Bresina, J. 1990. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. *Proc. of AAAI-90.* Boston, MA.

Drummond, M., and Kaelbling, L. 1990. Integrated Agent Architectures: Benchmark Tasks and Evaluation Metrics. In *Proc. of the DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA. pp. 408-411. Morgan-Kaufmann, San Mateo, CA.

Drummond, M. 1989. Situated Control Rules. *Proc. of Conference on Principles of Knowledge Representation & Reasoning.* Toronto, Canada.

Drummond, M., and Currie, K. 1989. Goal-ordering in Partially Ordered Plans. In *Proc. of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI.

Drummond, M. 1987. A Representation of Action and Belief for Automatic Planning Systems," *Reasoning about Actions and Plans*, M. Georgeff and A. Lansky, Eds.. Morgan Kauffman.

Fikes, R., Hart, P., and Nilsson, N. 1972. Learning and Executing Generalized Robot Plans. *AI Journal*, Vol 3, pp. 251–288.

Fox, M.S., and Sycara, K.P. 1990. The CORTES Project: A Unified Framework for Planning, Scheduling and Control. In *Proc. of the DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pp 412–421. Morgan Kaufmann, CA.

Gupta, A. 1987. Explanation-Based Failure Recovery. *Proc. of the Sixth National Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA.

Hanks, S. 1990. Projecting Plans for Uncertain Worlds. Yale University, CS Department, YALE/CSD/RR#756.

Kedar, S. T., Bresina, J. B. and Dent, C. L. 1991. The Blind Leading the Blind: Mutual Refinement of Approximate Theories. In *Proc. of the Eighth International Workshop on Machine Learning*, Evanston, IL.

Laird, J. E., and Rosenbloom, P.S. 1990. Integrating Execution, Planning, and Learning in Soar for External Environments. In *Proc. of the 8th National Conference on AI.* pp. 1022–1029.

Langley, P. and Drummond, M. 1990. Toward an Experimental Science of Planning. In *Proc. of the DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA. pp. 109-114. Morgan-Kaufmann, San Mateo, CA.

Le Pape, Claude. 1985. SOJA: A Daily Workshop Scheduling System. In *Proc. of the 5th Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Warwick, United Kingdom.

Levinson, Richard. 1990. Autonomous Reaction and Prediction with Dynamic Time Constraints. Stanford Spring Symposium. (Available as University of Maryland Technical Report SCR TR 90-45, edited by Jim Hendler.)

Maes, P., and Brooks, R.A. 1990. Learning to Coordinate Behaviors. In *Proc. of the 8th National Conference on AI.* pp. 796–802.

Mahadevan, S., and Connell, J. 1990. Automatic Programming of Behaviour-Based Robots Using Reinforcement Learning. Technical Report, IBM TJ Watson Research Center, Yorktown Heights, NY.

Minton, S. 1988a. Quantitative Results Concerning the Utility of Explanation-Based Learning. In *Proc. AAAI-88*, Saint Paul, MN, pp. 564-569.

Minton, S. 1988b. Learning Search Control Knowledge: An Explanation-based Approach. Kluwer Academic Publishers, Boston, Massachusetts (Also available as Carnegie-Mellon CS Tech. Report CMU-CS-88-133).

Mitchell, T. M. 1990. Becoming Increasingly Reactive. In *Proc. of the 8th National Conference on AI.* pp. 1051–1058.

Mitchell, T. M. and Keller, R. M. and Kedar-Cabelli, S. T. 1986. Explanation-Based Learning: A Unifying View; pp. 47 – 80, *Machine Learning*, Vol. 1, No. 1.

Mostow, J. and Bhatnagar, N. 1987. Failsafe – A Floor Planner that Uses EBG to Learn from its Failures. *Proc. IJCAI-87* Milan, Italy; pp. 249-255.

Newell, A. 1990. The Quest for Architectures for Integrated Intelligent Systems. In *Proc. of the DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pp 377–378. Morgan Kaufmann, CA.

Philips, A.B., & Bresina, J.L. 1991. *NASA TileWorld Manual* (NASA Technical Report TR-FIA-91-04). Moffett Field, CA: NASA Ames Research Center, Code FIA.

Ramadge, P. and Wonham, W. 1989. The Control of Discrete Event Systems. *Proc. of the IEEE.* Vol. 77, No. 1 (January). pp. 81–98.

Schoppers, M. 1987. Universal Plans for Reactive Robots in Unpredictable Environments. *Proc. of the Tenth International Conference on Artificial Intelligence.* pp. 1039–1046, Milan, Italy.

Sutton, R. 1990. Integrated Architectures For Learning, Planning, and Reacting Based on Approximating Dynamical Programming. In *Proc. of the 7th National Conference on Machine Learning*, Austin TX.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE Dates attached	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE Titles/Authors - Attached	5. FUNDING NUMBERS
6. AUTHOR(S)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Code FIA - Artificial Intelligence Research Branch Information Sciences Division	8. PERFORMING ORGANIZATION REPORT NUMBER Attached
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Nasa/Ames Research Center Moffett Field, CA. 94035-1000	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Available for Public Distribution Peter Friedland 5/14/92 BRANCH CHIEF	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Abstracts ATTACHED

14. SUBJECT TERMS	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT