Collected Notes from The Benchmarks and Metrics Workshop[1] (NASA Ames, June 25, 1990)

edited by

Mark E. Drummond
Sterling Federal Systems
NASA Ames Research Center
Mail Stop: 244-17
Moffett Field, CA 94035

Leslie P. Kaelbling
Stanley J. Rosenschein
Teleos Research
576 Middlefield Road
Palo Alto, CA 94301

March, 1991

[1]The organization of this workshop and the preparation of this report have been jointly supported by NASA and DARPA. DARPA co-funding has been provided by the Information Sciences Technology Office under DARPA Order 7382.

Contents

Introduction and Overview

An *integrated agent architecture* is a theory or paradigm by which one may design and program intelligent agents. An intelligent agent is a collection of sensors, computers, and effectors, structured in such a way that the sensors can measure conditions in the world, the computers can process the sensor information, and the effectors can take action in the world. Changes in the world realized by the effectors close the loop to the agent's sensors, necessitating further sensing, computation, and action by the agent.

In recent years there has been a proliferation of proposals in the AI literature for integrated agent architectures. Each architecture offers an approach to the general problem of constructing an integrated agent. Unfortunately, the ways in which one architecture might be considered better than another are not always clear.

There has been a growing realization that many of the positive and negative aspects of an architecture become apparent only when experimental evaluation is performed and that to progress as a discipline, we must develop rigorous experimental methods. In addition to the intrinsic intellectual interest of experimentation, rigorous performance evaluation of systems is also a crucial practical concern to our research sponsors. DARPA, NASA, and AFOSR (among others) are all actively searching for better ways of experimentally evaluating alternative approaches to building intelligent agents.

One tool for experimental evaluation involves testing systems on benchmark tasks in order to assess their relative performance. As part of a joint DARPA- and NASA-funded project, NASA-Ames and Teleos Research are carrying out a research effort to establish a set of benchmark tasks and evaluation metrics by which the performance of agent architectures may be determined. As part of this project, we held a workshop on *Benchmarks and Metrics* at the NASA Ames Research Center on June 25, 1990. The objective of the workshop was to foster early discussion on this important topic. We did not achieve a consensus of opinion, nor did we expect to.

This report collects together in one place some of the information that was exchanged at the workshop. This report includes an outline of the workshop, a list of the participants, notes taken on the white-board during open discussions, position papers/notes from some participants, and copies of slides used in the presentations.

Acknowledgements

The editors would like to thank Mel Montemerlo (NASA HQ), Lt Col Stephen E. Cross (DARPA), and Peter Friedland (NASA Ames) for making the workshop possible. Also, many thanks to Martha Del Alto for helping with the production of this report.

Workshop Outline

8:30 Coffee

9:00 Introduction

- General points.
- Domains and architectures of interest.
- Focus on characteristics of run-time system (*e.g.*, response time) vs. characteristics of development process (*e.g.*, ease of development.)
- Focus on external characteristics of system (*e.g.*, response time) vs. internal characteristics which may be architecture-specific (*e.g.*, number of subgoals generated.)
- How to (informally?) control for differences in underlying languages, compilation/interpretation environments, and machines.

9:30 How to specify a "benchmark task"

- What counts as a "task"? As a "metric"?
- Tasks and metrics must be chosen together.
- To specify task, need:
  - Environment description
  - Description of inputs & outputs of agent
  - Metric (Criteria for evaluating performance of agent)
- How should these be formulated (*e.g.*, formal, informal, via simulators, etc.)
- Compare and contrast agent benchmarks with standard benchmarks in other fields, *e.g.*, processor benchmarks, compiler benchmarks, etc.

10:30 Break

10:45 Three sample benchmark tasks/metrics and their attributes

- Presentation of strawman tasks in three domains:
  - *Tunable Benchmarks for Agent Evaluation* (M. Pollack)
  - *Benchmarks and Metrics for Mobile Agents* (R. Brooks)
  - *Requirements for Intelligent Monitoring Agents* (B. Hayes-Roth)
- Discussion will center on significant problem attributes exhibited by each of the strawman tasks (*e.g.*, time stress, uncertainty, run-time goals) and on objective metrics.

12:00 Lunch

1:00 Gap analysis

- Discussion will focus on the degree of match/mismatch between the tasks and metrics discussed in the morning and the specific tasks being used by workshop participants to evaluate their own work.
- The output will be a list of relevant task characteristics and a set of representative task instances embodying these characteristics, and possibly, some strategies and heuristics for generating and refining additional tasks and evaluation metrics.

3:15 Break

3:30 Standard robotic test platform

- Discussion of whether standard robotic platforms (*e.g.*, mobile robots, vision accelerators, etc.) should be made available to the research community and, if so, what their characteristics should be.
- How might the design of such standard platforms be established?
- How might platform development be funded?
- Who manages the distribution of the platforms?
- Is standardization premature? Will it stifle creativity?

5:00 End

6:00 Dinner

Workshop Participants

James Albus, NIST	albus@cme.nist.gov
Jim Antonisse, Mitre	antonisse@starbase.mitre.org
Hamid Berenji, NASA Ames	berenji@ptolemy.arc.nasa.gov
Pete Bonasso, Mitre	bonasso@starbase.mitre.org
John Bresina, NASA Ames	bresina@ptolemy.arc.nasa.gov
Rod Brooks, MIT	brooks@ai.mit.edu
David Chapman, Teleos Research	zvona@teleos.com
Steve Cross, DARPA	cross@vax.darpa.mil
Mark Drummond, NASA Ames	drummond@ptolemy.arc.nasa.gov
Jim Firby, JPL	firby@ai.jpl.nasa.gov
Carl Friedlander, ISX	cfriedla@isx.com
Armen Gabrielian, Thomson-CSF	tcipro!armen@unix.sri.com
Michael Genesereth, Stanford	mrg@sunburn.stanford.edu
Steve Hanks, U of Washington	hanks@cs.washington.edu
Barbara Hayes-Roth, Stanford	bhr@sumex-aim.stanford.edu
Jim Hendler, U Maryland	hendler@cs.umd.edu
Felix Ingrand, SRI	felix@ai.sri.com
Neil Jacobstein, Cimflex Teknowledge	njcaobst@teknowledge.com
Leslie Kaelbling, Teleos Research	leslie@teleos.com
Smadar Kedar, NASA Ames	kedar@ptolemy.arc.nasa.gov
John Laird, U of Michigan	laird@caen.engin.umich.edu
Pat Langley, NASA Ames	langley@ptolemy.arc.nasa.gov
Amy Lansky, NASA Ames	lansky@ptolemy.arc.nasa.gov
Rich Levinson, NASA Ames	rich@ptolemy.arc.nasa.gov
Dave Miller, JPL	dmiller@ai.jpl.nasa.gov
Steve Minton, NASA Ames	minton@ptolemy.arc.nasa.gov
Tom Mitchell, CMU	mitchell@daylily.learning.cs.cmu.edu
Andrew Philips, NASA Ames	abp@ptolemy.arc.nasa.gov
Martha Pollack, SRI	pollack@ai.sri.com
Paul Rosenbloom, ISI	rosenblo@venera.isi.edu
Stan Rosenschein, Teleos Research	stan@teleos.com
Marcel Schoppers, ADS	marcel@ads.com
Reid Simmons, CMU	reid.simmons@freesia.learning.cs.cmu.edu
Keith Swanson, NASA Ames	swanson@ptolemy.arc.nasa.gov
David Thompson, NASA Ames	det@ptolemy.arc.nasa.gov
David Tseng, Hughes	tseng@aic.hrl.hac.com
Dan Weld, U of Washington	weld@cs.washington.edu

White-Board Lists

This section contains three sets of items related to the evaluation of intelligent agent architectures:

(1) tasks that could be used to test proposed architectures;

(2) key attributes of tasks, or dimensions along which tasks can vary; and,

(3) requirements for hardware that could serve as a common experimental platform.

Lists (1) and (3) were generated by workshop participants in brainstorming sessions during the course of the workshop; list (2) was synthesized by the authors immediately after the workshop from ideas inspired by general discussion. In the spirit of the workshop, these lists are intended to be suggestive rather than comprehensive.

Possible Tasks

- Indoor robotic navigation and delivery.
- Outdoor survey and collection.
- Automated indoor tour guide.
- Multiple coordinated robots (with and without explicit communication).
- Satellite servicing (ORU replacement with verification).
- Find and assemble: definitions and instructions at runtime.
- Assemble, diagnose and repair electromechanical devices.
- Outdoor robots – real-time terrain reasoning.
- Cleaning and deburring machined parts.
- Space station assemply, maintenance, refueling.
- Process scheduling.
- Space shuttle refurbishment – scheduling, planning, diagnosis, repair.
- Cleaning teenager's room.
- Nurse's associate.

- Building construction.
- Data analysis planning (Earth Observing System).
- SIMNET player (robotic).
- Traffic world.
- Aircraft emergency procedures.
- Planning a tour in dynamic environment.
- Vehicle Driving.
- Rendezvous docking operations.
- Road grading.
- Beacon emplacement (to define landing areas).
- Structure protection (against external influences).
- Aerobrake assembly.
- Toxic waste handling and disposal.
- Ventilator management.

Possible Task Attributes

Resource Management. Does the task have properties pertaining to metric time and continuous quantities? If so, the problem may take on the character of a classical optimization problem.

Geometric and Temporal Reasoning. Does the task involve extensive geometry or reasoning about activities over time? These kinds of reasoning may require specialized representations.

Deadlines. Does the task impose absolute deadlines for goal satisfaction, or does the utility of goal satisfaction vary continuously with time?

Opportunity for Learning. Is the task specification complete at the beginning of the agent's execution? If not, the agent must be able to acquire knowledge about its environment.

Multiple Agency. Does the task require the definition of a community of interacting agents? Such tasks might require complex communication protocols or reasoning about the internal states of other agents.

Informability. Can the agent be presented with explicit goals and facts about the world during the course of its execution?

Dynamic Environment. Does the agent's world change over time independent of the actions the agent takes? How predictable are the dynamics of the world?

Amount of Knowledge. How much *a priori* knowledge is available to be used by the agent? Some domains, such as medical diagnosis, require the assimilation of a large amount of domain knowledge.

Reliability of Sensors and Effectors. In some task domains, sensors and effectors are completely reliable; in others, the main difficulty lies in accurately integrating data from a number of highly unreliable sensors and achieving robust overall behavior through the use of unreliable effectors.

Requirements for a Common Hardware Platform

- Manipulator.
- High speed data and control link.
- Easy to control from various machines.
- Payload capability.
- Modularity.
- Comes with a simulator.
- Standard bus card cage.
- Hierarchical sensor & effector software (felexible software library).
- Reliability.
- On-Platform debugging facility.
- Safety.
- Turnkey functionality.
- Affordable.

Position Papers and Notes

Some workshop participants contributed written position papers, and others provided short working notes. The following pages reproduce these papers and notes, with the authors' collective permission.

It is important to understand the status of the following material. The Ames meeting on June 25 was intended to be a *bona fide* workshop, and participants were encouraged to submit written (but preliminary!) accounts of ideas, issues, and concerns. To try to capture the spirit of the workshop, we have included some of this material in the following pages. Thus, these notes should not be read as a proceedings of the workshop, but rather, as a workshop "snapshot".

Please note that some authors have since expanded upon their workshop contributions. In particular, Paul Cohen has written a paper entitled "A Survey of the Eighth National Conference on Artificial Intelligence: Pulling Together or Pulling Apart" (AI Magazine, Spring 1991). Pat Langley, Leslie Kaelbling, and Mark Drummond have also written about related issues, and their papers can be found in the Proceedings of the DARPA workshop on innovative approaches to planning, scheduling and control, held in San Diego, CA, from November 5th through the 8th, 1990. The Proceedings are available from Morgan-Kaufmann, San Mateo, CA.

Authors and Titles

THE DIVERSITY METRIC FOR INTELLIGENT SYSTEMS
Proposed at the Benchmark and Metric Workshop, NASA Ames, 25 Jun 90
Revision #1, 18 Jul 90

Jaime Carbonell, Tom Mitchell, Allen Newell
School of Computer Science, Carnegie Mellon University

Preface: The forte of intelligent systems is the ability to deal with a broad diversity of tasks. We seek a measure of this ability to handle diversity.

Proposal:

A **combo** is a **system** with a **developer**, where:
The **system** is the system to be measured.
The **developer** is a human with the expert skills and knowledge to operate, command, guide, instruct, reprogram and develop the **system** (hereafter, **modify** the system).

A **task domain** (**D**) is a set of **tasks**, where:
Task accomplishment can be specified to some **competency level** (**CL**).
The **domain** can be described coherently to a **developer**.

The **completion time** (**T**) of a **combo** for a **task** from domain **D** is the time required by the **combo** to achieve the **competence level** specified for the **task**, where:
There can be prior knowledge of the domain **D**.
There is no prior knowledge of the **task** and **CL** before the clock starts.
The **developer** can **modify** the **system** at will.
The clock stops when the **system** performs the **task** by itself.

The **diversity** of a system for a **domain D** is the set of tasks in **D** for which **completion times** are acceptable, where:
Acceptability is a parameter set to reflect the context of use in which the tasks occur.

Intelligent systems are to be evaluated in terms of the **diversity** they can achieve.

Notes:

1. The key idea is to permit development and modification, but to include this time in the measure of the system's effectiveness.
2. Modification time should not be separated from system-performance time, because the system's capabilities for flexibility may help as much in modification as in performance.
3. The system external interfaces (natural language, robotic, formal specification languages) play a critical role in the speed of its modification and the domains it can attempt. They are not fixed as part of the task specification, because the **combo** may modify them.
4. This metric assigns the full time cost of modification to a single performance, which is appropriate for many domains but not all; modification could be amortized over several performances.
5. Intelligent systems will be competitive for domains whose tasks pose significant requirements for assimilation, learning, adaptation and modification; specialized systems will be competitive for domains whose tasks are stable with prespecified variability.
6. The **combo** situation permits assessment before a system's adaptive capability is advanced enough to focus exclusively on measuring autonomous behavior.
7. The diversity metric is equally applicable to reuseability, maintainability and adaptability in software-engineering.
8. Interesting domains include: acquiring a new task domain, acquiring a major new functional capability, incrementing functionality, coping with a new environmental feature, accepting a new communication convention, exploiting a relaxation of time pressure, etc.

On choosing domains for agents

David Chapman

Teleos Research
576 Middlefield Road
Palo Alto, CA 94301
(415)328-8879

Position paper prepared for the
Workshop on Benchmarks and Metrics
NASA Ames, June 25, 1990.

Abstract

Much recent work has addressed the problems of uncertain, real-time domains. This paper argues that these *negative* characterizations of domains are inadequate: they make an already difficult problem harder. Focussing on these characteristics leads one into intractable search problems and unrealistic noise models. We need rather to focus on the *positive* attributes of domains that make action possible. I analyze the success of Sonja and Pengi in these terms. I propose that research should seek new facilitating features and architectures that can exploit them.

This research was supported by the Air Force Office of Scientific Research under contract F49620-89-C-0055.

1 Negative characterizations of domains

In our original paper on Pengi [1], Phil Agre and I characterized interesting domains as *complex, uncertain,* and *real time* and described Pengi's domain Pengo as one such. We focussed on these attributes of the domain to make the point that available theories of activity (based on planning) were unable to deal with important characteristics of real-world tasks.

In the past few years, uncertainty and real-time response have become central issues for theories of activity. (Complexity has been less studied; I'll come back to this.) This paper argues that while it is indeed important that real-world domains have these characteristics, focussing on them in isolation is a mistake.

1.1 Combinatorics

Planning problems are hard because of combinatorics. Adding uncertainty makes them much worse.

If all you know about a domain is that it is uncertain and real-time, all you can do is brute-force search. There's nothing grab onto to work with, no constraint to exploit. The search problems that come up in planning are unpleasantly intractable. What's worse, they aren't interesting. Without additional imposed structure, "planning is just computation" [2]; we can't expect to do anything clever.

Few researchers have tried to address the issue of domain complexity. The reason, I think, is that complexity is the feature that makes search infeasible. As long as activity research continues to focus on search-based techniques, the complexity of real domains must continue to be ignored.

Toy domains are, I believe, *too hard* because they are *too simple.* Complexity is a problem for search-based techniques, but it can be a resource for others. Simplifying a domain removes all the useful structure. (More about this later.)

The combinatorics of planning are the combinatorics of the problem, not of the solution. In other words, nontraditional solutions to the traditional planning problem can't work either. Since creatures do act sensibly in the real world, it must be that the *problem* is wrong, and too hard. It's wrong because it's too general. Neither people nor computers can act effectively in *arbitrary* uncertain, real-time domains.

1.2 Uncertainty, noise, randomness, and emergent structure

It's common these days to work in simulated domains in which certain aspects are subject to deliberately introduced pseudorandom variation. This is intended to model the observation that real-world domains often vary unpredictably.

I believe that deliberately introducing simple forms of randomness is misleading and results in qualitatively different agent/world dynamics than the sorts of uncertainty that are prevalent in the real world.

Although the real world may contain some genuine randomness due to quantum noise, this is not the major source of uncertainty in the macroscopic domains. Uncertainty instead derives from lack of knowledge. The world is much too *complex* to represent all of it, even if you could find out about all of it, and too complex to simulate even if you could represent it. The world looks much more random, when viewed through the perspective of any one agent, than it actually is.

Why does this matter? It matters because beneath the apparent randomness of any particular real-world task there lies a great deal of *structure* which sensible agents rely on. The "noise" in real domains is not distributed uniformly, but in complex patterns which can—and must—be exploited to get useful work done. This exploitation may come from a bottom-up learning process or by explicit design by a theorist who also has observed and figured out how to make use of patterns whose causes are obscure.

By contrast, the noise that is added to simulated worlds is typically distributed uniformly or in some other simple way. This is because uncertainty is seen as a difficulty to be overcome, rather than a resource to exploit. If randomness is just an obstruction, it doesn't matter if it's just uniform. And indeed no use one can make of uniformly distributed noise; all you can do is work around it. All it does is make a bad scheduling problem worse. But if structured variation is a crucial resource for effective action, artificial noise results in an impoverished, unrealistically difficult world.

2 Positive characterizations of domains

In addition to explaining what makes the real world *hard* to act in, we need to say why it is *possible*. That is, we need positive characterizations of domains as well as negative ones. How can we choose facilitating characteristics in a principled way? I'll begin to answer this question by looking at some characteristics that successful existing systems exploit.

2.1 Why Sonja and Pengi work

Sonja [4] and Pengi [1] operate effectively in complex, uncertain, real-time domains (Amazon and Pengo). What features of these domains do they exploit?

It is impossible to answer this question definitely without extensive cross-domain research. I have some ideas, though. Sonja and Pengi work by means of visual routines that analyze the spatial configurations of the domain objects to extract task-relevant properties. For example, Sonja looks to see whether it has a clear shot at a monster. What domain features enable such routines?

- In these domains, the effects of the agent's actions and of other processes are both spatially constrained. Amazons, ghosts, and fireballs move slowly enough that they traverse only small distances over the cycle time of the agent.
- Most of what is in any situation is irrelevant to any given task. Although any ice cube may become important at some point in a Pengo game, all but a few are uninteresting at any given time.
- Moreover, it is possible to find the relevant objects easily. This property along with the former two make it possible to avoid representing most of the domain, and thereby to avoid the combinatorics that the domain's complexity would otherwise engender. Finding relevant objects depends on other properties:
  - It is possible to perceive almost everything relevant with minimal effort. Sonja and Pengi have visual bandwidth limitations which model human visual bandwidth limitations, and these prevent the systems from looking at more than a tiny fraction of the visually presented scene at once. Moreover, in Amazon (but not Pengo) most domain objects are occluded at any given time. Nonetheless, it is usually possible in both domains to use visual search routines to find the most important objects in a situation quickly relative to the pace of the game.
  - Interesting objects are conspicuous (in terms of available perceptual primitives). For example, they may be moving, whereas uninteresting objects are mostly stationary; or they may distinctively colored, or distinctively placed in relationship to other interesting objects.
- Visually identifiable properties of objects strongly constrain their future behavior. Thus it is possible to reason about the future by means of geometrical computations (efficiently implemented in the visual system) rather than by deduction from frame axioms.

2.2 Future directions

The positive properties I enumerated in the last section are not true of every domain. However, they are mostly true of many domains most of the time. This makes them reasonable properties for an architecture to depend on.

More generally, interesting positive properties should not be domain-specific, or we will only learn about one domain. On the other hand, we should not expect to find positive properties that hold of *every* domain of interest. Rather, we should enumerate many properties that are often true, and design architectures that can flexibly exploit those that are available and that degrade gracefully as facilitating features are removed.

There are several positive properties of domains that I suspect are true of most domains of interest, that I think important to human performance, and that I think we ought to investigate further.

- In many or most domains, activity is inextricably intertwined with perception. The sort of input an "action component" gets may be crucial to determining its architecture. Artificial domains that abstract away from perception may lead us down blind alleys. As I have argued elsewhere [3], I believe that the "simplification" of studying activity in isolation from perception makes the problem harder, not easier. The extraordinary richness of perception makes action easy.

- Real domains typically are regular or routine in the sense that a tiny subset of the logically or physically possible sorts of courses of events actually transpire. These regularities arise from complex interactions between domain processes. These processes and their interactions are typically unknown to agents acting in the domain, but in many cases are key to effective performance. How do agents come to exploit structure whose causes they don't understand?

- Typically people learn new tasks by trying the easy cases first. We need to understand how an agent can select its own training sequence and what properties of a domain make that easy. For example, a domain may make it perceptually simple to determine how difficult a problem instance is. A domain also has to make it possible for an agent to ignore or finesse the hard cases at first.

- Typically people learn new tasks by *legitimate peripheral participation* [5], i. e. by performing the task first as an assistant and gradually taking over more and more responsibility. What sorts of domains and tasks make legitimate peripheral participation possible, and how does this participation enable an agent to gradually take on more responsibility?

3 Summary

- Characterizing domains in solely negative terms leads to intractable and unrealistic pseudoproblems.

- We should, instead, look for the facilitating domain characteristics that make effective action possible.

- Such characteristics should not be domain-specific, but may also not hold of all domains all the time.

- Accordingly, no single domain can serve as an accurate benchmark. An architecture should be evaluated according to its versatility. We need a suite of benchmarks which are chosen to exhibit a range both of difficulties and resources.
- Domain complexity has been neglected as a source both of difficulty and opportunity.

Acknowledgements

This paper was inspired by a conversation with Martha Pollack. Many of the ideas in it, particularly the importance of positive characterizations of domains, are due to Phil Agre, who also read a draft. Leslie Kaelbling and Stan Rosenschein encouraged me to write it, and Stan also commented on a draft.

References

[1] Philip E. Agre and David Chapman, "Pengi: An Implementation of a Theory of Activity," AAAI-87.

[2] David Chapman, "Planning for Conjunctive Goals." *Artificial Intelligence*, **32** (1987) pp. 333-377.

[3] David Chapman, "Penguins *Can* Make Cake." *AI Magazine*, Vol. 10, No. 4 (Winter 1989), pp. 45–50.

[4] David Chapman, *Vision, Instruction, and Action.* MIT AI TR 1204, April 1990.

[5] Jean Lave and Etienne Wenger, *Situated Learning: Legitimate Peripheral Participation.* Institute for Research on Learning Report No. IRL90-0013, February 1990.

Benchmarks Are Not Enough
Evaluation Metrics Depend on the Hypothesis

Paul R. Cohen and **Adele E. Howe**
Experimental Knowledge Systems Laboratory
Lederle Graduate Research Center
Department of Computer and Information Science
University of Massachusetts, Amherst

cohen@cs.umass.edu
howe@cs.umass.edu

June 1990

A couple of weeks ago we got an invitation: "to participate in the Benchmarks and Metrics Workshop (BMW), a one-day working meeting on evaluating agent architectures. The purpose of the meeting is to generate ideas on possible benchmark problems and evaluation metrics." Unable to attend, we console ourselves by writing this.

Our concern is that evaluation has become synonymous with performance evaluation. This leads to what John McCarthy is reputed to have called "look Ma, no hands" demonstrations. Benchmarks are a small step forward: they standardize what is being demonstrated, but they also perpetuate the view that evaluation is no more than demonstration of performance.

When we build AI systems as demonstrations, guided by benchmarks, we tend to aim for the benchmark and eliminate behaviors that, in the light of the benchmarks, we interpret as "bugs." As autonomous agents become more interesting, this style of research turns a blind (or prejudiced) eye to phenomena that lead to novel hypotheses. We end up building AI systems that do what they are expected to do. The literature is full of papers that assert, "We need X, here's how we expect to provide X, and (later) here's a demonstration of X." Lenat and Feigenbaum put it this way:

> "If one builds programs that cannot possibly surprise him/her, then one is using the computer either (a) as an engineering workhorse, or (b) as a fancy sort of word processor (to help articulate one's hypothesis), or, at worst, (c) as a (self-) deceptive device masquerading as an experiment. ... The most profitable way to investigate AI is to embody our hypotheses in programs, and gather data by running the programs. ... Progress depends on the experiments being able to falsify our hypotheses."

Hypotheses are already very rare in AI. The general form of a hypothesis is "X is sufficient to produce Y." But unlike hypothetico-deductive science, we rarely show the necessity of one hypothesis vis a vis mutually exclusive alternatives. Our principal mode is to accept the null hypothesis: to accrue demonstrations. It is difficult to see how instituting benchmarks—with their emphasis on demonstrating performance—will change this methodological stance.

Let us accept that AI research should be driven not by benchmarks but by hypotheses about why AI systems behave as they do in particular environments. What does this imply about evaluation metrics? There are two kinds of metrics: measures of performance, and measures of those factors we hypothesize are causally responsible for levels of performance (often called dependent and independent measures, respectively). Experiments test these causal hypotheses. In the discussions at this workshop, it seems essential to focus on a methodology to get at these causal hypotheses—on dependent and independent measures, both. Benchmarks *might* be designed to anticipate causal hypotheses, but failing that, they certainly should not be accepted in lieu of hypotheses.

Paul R. Cohen and Adele Howe. 1988. Toward AI Research Methodology: Three Case Studies. *IEEE Transactions on Systems, Man and Cybernetics.* Vol. 19, No. 3. pp. 634–646.

Paul R. Cohen and Adele E. Howe. 1988. How Evaluation Guides AI Research. *AI Magazine*, Winter, 1988. Vol. 9, No. 4, pp. 35–43.

Paul. R. Cohen. Evaluation and Case-based Reasoning. *Proceedings of the Second Annual Workshop on Case-based Reasoning*, Pensacola Beach, FL. May 30–June 2, 1989. pp. 168–172.

Lenat, D. and Feigenbaum, E.A., On the thresholds of knowledge. *Proceedings of IJCAI 10.* pp. 1173 -1182.

Phoenix: A Testbed for Shared Planning Research[2]

David M. Hart and **Paul R. Cohen**
Experimental Knowledge Systems Laboratory
Computer and Information Science
University of Massachusetts
Amherst, MA 01003

dhart@cs.umass.edu
cohen@cs.umass.edu

June 1990

Abstract

We describe an instrumented simulation testbed called Phoenix that we have developed for a complex, dynamic environment—forest fire-fighting. Phoenix has many built-in features to support the evaluation of autonomous planning agents. Thus, the Phoenix testbed bears consideration as a paradigmatic environment for current planning research.

[2]This research has been supported 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 Air Force Office of Scientific Research contract #49620-89-C-0121; and ONR contract #N00014-88-K-004.

1 Introduction

We have seen a revival in planning research in the last few years as attention has turned to AI systems that operate in complex, dynamic environments. A burgeoning number of task domains and a variety of new and old planning approaches are being explored. Several questions arise from such diversity: how do we evaluate this work? can we compare one approach to another? what characteristics do we require in a task environment, and which environments have those characteristics? In this note we describe an instrumented simulation testbed called Phoenix that bears consideration as one of a set of paradigmatic problems in current planning research.

Several years ago we chose a task domain that has many of the characteristics we associate with complex, dynamic environments—forest fire-fighting. Our goal has been to design autonomous agents for this environment. Our methodology requires empirical analysis of the environment and of the behaviors of the agents we design. To support this analysis, we built the Phoenix testbed [1] and a variety of tools for development and experimentation. Sections 2 and 3 describe the Phoenix task domain and the characteristics that make it paradigmatic. Section 4 describes the features of the testbed that support development and evaluation. Section 5 shows the layered modularity of the system and how other researchers could use part or all of it to test their own approaches to planning. Section 6 gives a partial list of current and potential areas of research in Phoenix to illustrate the richness of this task domain.

2 The Task Domain: Controlling Simulated Forest Fires

The Phoenix task is to control simulated forest fires by deploying simulated bulldozers, helicopters, fuel carriers, and other objects. The Phoenix environment simulates fires in Yellowstone National Park, for which we have constructed a representation from Defense Mapping Agency data. As the simulation runs, the user views fires spreading and agents moving through a topographical map of the park that shows elevations, ground cover, static features such as roads and rivers, and dynamic features such as fireline.

Fires spread in irregular shapes, at variable rates, determined by ground cover, elevation, moisture content, wind speed and direction, and natural boundaries. For example, fires spread more quickly in brush than in mature forest, are pushed in the direction of the wind and uphill, burn dry fuel more readily, and so on. These conditions also determine the probability that the fire will jump fireline and natural boundaries.

Fires are fought by removing one or more of the things that keep them burning: fuel, heat, and air. Cutting fireline removes fuel. Dropping water and flame retardant removes heat and air, respectively. In major forest fires, controlled backfires are set to burn areas in the path of wildfires and thus deny them fuel. Huge "project" fires, like those in Yellowstone several years ago, are managed by many geographically dispersed firebosses and hundreds of firefighters. The current Phoenix planner is a bit more modest. One fireboss directs a number of bulldozers to cut line near the fire boundary under moderate conditions, or at some distance from the fire when it is spreading quickly.

3 Characteristics of the Fire-fighting Domain

Several characteristics of this environment constrain the design of agents, and the behaviors agents must display to succeed at their tasks. The fire environment is dynamic because everything changes: wind speed and direction, humidity, fuel type, the size and intensity of the fire, the availability and position of fire-fighting objects, the quantity and quality of information about the fire, and so on. The environment is ongoing in the sense that there isn't a single, well-defined problem to be solved, after which the system quits; but, rather, a continuous flow of problems, most of which were unanticipated. The environment is real-time in the sense that the fire "sets the pace" to which the agent must adapt. The agent's actions, including thinking, take time, and during that time, the environment is changing. Additionally, agents must be able to perceive changes in the environment, either directly through their own senses or indirectly through communication with other agents.

The environment is unpredictable because fires may erupt at any time and any place, because weather conditions can change abruptly, and because agents may encounter unexpected terrain, or fire, or other agents as they carry out plans. An agent must respond to unexpected outcomes of its own actions (including the actions taking more or less time than expected) and to changes in the state of the world.

The fact that events happen at different scales in the Phoenix environment has profound consequences for agent design. Temporal scales range from seconds to days, spatial scales from meters to kilometers. Agents' planning activities also take place at disparate scales; for example, a bulldozer agent must react quickly enough to follow a road without straying due to momentary inattention, and must also plan several hours of fire- fighting activity, and must do both within the time constraints imposed by the environment. (Notably, Phoenix agents are hybrid reactive/deliberative planners.)

The Phoenix environment is spatially distributed, and individual agents have only limited, local knowledge of the environment. Moreover, most fires are too big for a single agent to control. These constraints dictate multi-agent, distributed solutions to planning problems.

4 The Phoenix Testbed

The Phoenix testbed simulates the forest fire environment. It uses map structures to represent the forest, and a discrete event simulator to coordinate tasks that effect spreading fires, changes in the weather, and the actions of agents. The agents we design interact with this simulated world through a prescribed interface for sensors and effectors. Agent design and planning research are done in layers of the system built on top of the testbed (see Section 5).

The advantages of simulated environments are that they can be instrumented and controlled, and provide variety–all essential characteristics for experimental research. The Phoenix testbed has many features that facilitate experiments. It provides development tools for implementors, instrumentation for evaluation and analysis, baselines and baseline scenarios for benchmarking, and an interface for managing large experiments.

4.1 Development tools

The testbed includes tools to help developers implement and debug fire-fighting agents. The graphic interface (the map of the park) is highly interactive, allowing the user to zoom in and out at different resolutions, watch agents' movements and fire-fighting activities, determine static and dynamic features at any point in the map, and "see" the view of each individual agent (each agent's knowledge about dynamic features in the environment is determined by what it has perceived directly and what has been communicated from other agents). A desktop interface allows the user to interrogate each agent's memory structures, which are built from a generic frame system. A grapher and frame-editor allow the user to browse through memory structures and alter values for debugging.

4.2 Instrumentation

Phoenix is instrumented at three levels: the implementation level of system performance, the solution level of planning and agent designs, and the domain level of fire-fighting. Examples follow:

The implementation level includes metering to measure run time, cpu time , disk wait time, time since last run , idle time, and utilization, each graphed against time; as well as an interface to the Explorer performance metering that gives, for each function, the number of calls , average run time, total run time, real time, memory allocation, page faults, and so on.

The solution level is characterized by statistics on cpu utilization by the cognitive component of each agent, showing the profile of real-time response; for example, the latency between when actions on the timeline become available for execution and when they are executed. Other metering provides metrics on sensor and effector usage, and on reflexes.

The domain level measures performance in domain terms; for example, the amount and type of forest burned, the number of houses burned, and the number of agents that perish. Resource allocation is currently measured by the amount and type of agents employed to fight the fire, gasoline consumed, fireline cut, distance traveled, and time required to contain the fire.

4.3 Baselines and baseline scenarios

We have developed baseline scenarios for several situations that are both characteristic of this domain and require timely response to environmental changes. As an example, one scenario involves a single fire whose profile changes dramatically due to an increase in wind speed and a change in wind direction, so that the nature of the threat changes from the potential loss of forest to loss of populated areas. Another scenario presents the problem solver with multiple fires and limited fire-fighting resources which must be managed efficiently to prevent one or more fires from spreading out of control. By limiting the number of fire-fighting agents available, and limiting the amount of fuel each can carry (requiring them to refuel periodically), this scenario forces the problem solver to allocate its resources wisely in order to control all the fires.

We are collecting a wide range of baseline data about the simulation environment that we will use to build our agents' planning knowledge- bases. This includes data about fire spread rates, agent movement, reaction and thinking time in agents, and effectiveness of fire-fighting strategies. Baseline data are also used to evaluate the performance of our agents.

A scripting capability allows the user to create and store scenarios. Scripts give control over environmental factors such as when and where fires start, wind characteristics, and the resources available for fire- fighting (how many agents of each type, what are their speeds, fuel capacities, fields of view). The timing of environmental changes is specified in scripts, allowing the user to control when events occur in the simulation for testing purposes. Instrumentation functions can be run within scripts.

4.4 Experimental interface

Scripts are part of a suite of data collection and analysis programs, which also includes facilities for manipulating data, a statistical package, and a graphing package. With these tools we have designed, executed, and analyzed the data from several large experiments involving—to date—over two thousand fire-fighting episodes.

5 Phoenix System Components

Phoenix runs on TI Explorers (color or B&W) and MicroExplorers, and is packaged into one system that includes all non-standard (non-proprietary) support code we use to run it. Part or all of the system can be provided to other laboratories and research groups on tape as a TI Load Band along with supporting documentation for the testbed, on-line help facilities, and annotated Lisp source code.

The Phoenix system comprises five levels of software:

1. DES — the discrete event simulator kernel. This handles the low-level scheduling of agent and environment processes. Agent processes include sensors, effectors, reflexes, and a variety of cognitive actions. Environment processes include fire, wind, and weather. The DES provides an illusion of simultaneity for multiple agents and multiple fires.
2. Map —this level contains the data structures that represent the current state of the world as perceived by agents, as well as "the world as it really is." Color graphics representations of the world are generated from these data structures.
3. Basic agent architecture— a "skeleton" architecture from which agents, such as bulldozers, helicopters, and firebosses are created. The agent architecture provides for sensors, effectors, reflexes, and a variety of styles of planning.
4. Phoenix agents— the agents we have designed (and are designing) for our own experiments.
5. Phoenix organization—currently we have a hierarchical organization of Phoenix agents, in which one fireboss directs (but does not control) multiple agents. Each Phoenix agent is autonomous and interprets the fireboss's directions in its local context, while the fireboss maintains a global view.

Phoenix is modularized so that other researchers can work with all or part of it. The first two levels, above, comprise the fire simulation testbed; researchers interested in designing and implementing their own agent architectures could experiment with them in this testbed. The next level is a generic agent architecture shell–a set of functional components common to all agents. Code to interface these components with the testbed is included with this level, so that researchers interested in working with our functional decomposition of agent capabilities need only instantiate the shells for components (sensors, effectors, reflexes, and cognitive capabilities) with their own designs. The fourth level includes our designs of these components, providing a specific agent architecture that is distinctive primarily for the planning style used in the cognitive component (skeletal planning with delayed commitment to specific actions). Researchers interested in using our planning style and agent architecture could work with the first four levels, creating their own agent types and organizing them according to their research interests. The fifth level is the organization of fire fighting agents. Researchers interested in working with our solution to problem solving in this domain could use all five levels to replicate and/or extend our work

6 Research Issues in Phoenix

The Phoenix environment presents a rich variety of research issues. We are just scratching the surface with the ones we've tackled, which include: modeling the environment and agent architecture [3], building adaptive capabilities into the planner based on error recovery [6,7], sophisticated monitoring and control of plan execution [4], and real-time problem-solving [2,5]. Future areas of research we have identified are resource management, situation assessment, different protocols for the integration of reactive and deliberative control, and different types of learning.

7 Conclusion

Phoenix is an instrumented testbed that simulates a complex, dynamic environment. The task domain, forest fire-fighting, has many of the characteristics that define these environments. The testbed has many features designed to support empirical analysis, and is modularized for use by other researchers. This environment presents a variety of open research questions.

References

1. Paul R. Cohen, Michael L. Greenberg, David M. Hart, and A.E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3): 32-48.

2. Paul R. Cohen, A.E. Howe, and David M. Hart. Intelligent real-time problem solving: Issues and examples. *Intelligent Real-Time Problem Solving: Workshop Report*, edited by Lee D. Erman, Santa Cruz, CA, November 8-9, 1989, pgs IX-1–IX-34.

3. Paul R. Cohen. Designing and analyzing the Phoenix planner with models of the interactions between Phoenix agents and the Phoenix environment. Technical Report #90-22, Dept. of Computer and Information Science, University of Massachusetts, Amherst, March, 1990.

4. David M. Hart, Scott D. Anderson, and Paul R. Cohen. Envelopes as a vehicle for improving the efficiency of plan execution. Technical Report #90-21, Dept. of Computer and Information Science, University of Massachusetts, Amherst, March, 1990.

5. Adele E. Howe, David M. Hart, and Paul R. Cohen. Addressing real- time constraints in the design of autonomous agents. *Real-Time Systems*, 2(1/2): 81-97.

6. Adele E. Howe. Integrating adaptation with planning to improve behavior in unpredictable environments. In *Working Notes of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*, Palo Alto, CA, March 1990.

7. Adele E. Howe and Paul R. Cohen. Responding to Environmental Change. Technical Report #90-23, Dept. of Computer and Information Science, University of Massachusetts, Amherst, March, 1990.

Characteristics of Tasks for Intelligent Agent Benchmarks

John E. Laird
Artificial Intelligence Laboratory
The University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109-2110
laird@caen.engin.umich.edu

June 22, 1990

1 Introduction

The purpose of this document is to lay out the space of task characteristics that should be covered by a suite of benchmark tasks for an intelligent agent. We assume that a task can be characterized by a task environment (such as a room with blocks), a set of goals (stack the blocks in the corner), and an agent consisting of a perceptual system (transducer of energy in the environment into information, such as a camera), a motor system (a transducer of information into energy/action in the environment, such as an arm), and a cognitive system (processor of information). The task environment may be real or simulated, with corresponding real or simulated perceptual and motor systems for the agent.

A benchmark would consist of a specification of task environment and possible constraints on the other components. For example, a benchmark might be just a task environment and a set of goals, where all of the other components are free variables that are under control of the system designer. Another possibility is that the benchmark would be the task environment, the goals, plus predefined perceptual and motor systems (such as those provided by a specific mobile robot). There could even be constraints on the cognitive system, such as that it must fit in a certain amount of memory or use only a prespecified set of domain knowledge.

2 Characteristics of the Environment

1. Dynamics of the environment
   (a) changes over time, independent of the agent
   (b) different external processes have different dynamics
   (c) synchronous vs. asynchronous with agent
   (d) predictable vs. unpredictable
2. Interaction between the environment and the agent
   (a) hinder vs. help the agent's goal achievement
   (b) recoverable vs. unrecoverable failures
   (c) tools can change interaction between agent environment
      i. provide transduction of environment to agent or vice versa.
      ii. can improve perceptual sensitivity, accuracy, precision, etc.
      iii. can improve action effectiveness, accuracy, precision, etc.
   (d) other agents may be in environment
      i. hinder agent goal achievement
      ii. help agent goal achievement
      iii. have own goals that are consistent or inconsistent with the agent

3 Characteristics of the Agent's Goals

1. multiple goals
2. interacting goals
3. concurrent goals
4. time dependent goals
5. one-shot vs. continuous vs. cyclic goals
6. hard vs. soft constraints on goal achievement
7. internal vs. external sources for goals

4 Characteristics of Perceptual System

1. sensitivity of sensor
   (which forms and ranges of energy)
2. completeness of sensor
3. amount of data from sensor
4. speed of sensor
   (timeliness of data)
5. precision of sensor
6. accuracy of sensor
7. active vs. passive sensor
8. fixed vs. manipulable sensor

5 Characteristics of Motor System

1. impact on environment
   (how effector changes environment)
2. physical range of effectors
   (no action at a distance)
3. degrees of freedom of effectors
4. concurrency of action
   (sequential vs. parallel action)
5. dynamics of effector
   (speed of action)
6. predictability of action
7. precision of action
8. accuracy of action
9. feedback of effect of action

6 Characteristics of Cognition

1. speed of processing
2. online vs. offline processing
3. memory limitations
4. access times for different memory structures
5. initial knowledge
6. correctness of initial knowledge
7. types of knowledge (episodic, procedural, semantic)
8. indirect sources of knowledge about environment (trainer, map, encyclopedia, data base)
9. ability to improve knowledge through experience
10. autonomy (can not be restarted by an external agent)

Acknowledgments

Thanks to Ed Durfee, Allen Newell, Paul Rosenbloom, and the Michigan Soar research group.

The Experimental Study of Intelligent Agents

Pat Langley
AI Research Branch, MS: 244-17
NASA Ames Research Center
Moffett Field, CA 94035

To evaluate any method or system, one needs some measure(s) of its behavior. In most experiments, these will be the dependent variables. One obvious measure concerns the quality of the agent's behavior, such as the length of solution path or the total energy expended. In some cases, the agent may be unable to accomplish a task or solve a problem, and one can also measure the percentage of solved problems. A third class of dependent variables involves the amount of time or effort spent in generating a plan or behavior (though in resource-limited situations, this may also be controlled).

Evaluation also acknowledges that there are different approaches to the same problem, and this constitutes a major independent variable in experimental studies. One can run different methods or systems on the same task and measure their relative behavior along one or more dependent dimensions. The goal here is not to hold a competition but to increase understanding. Nor need one compare entire systems against each other; one can also lesion specific components or vary parameters to determine their effects on behavior.

Seldom will one system always appear superior to another, and this leads naturally to the idea of identifying the conditions under which one approach performs better than another. Real-world problems may have practical import, but they provide little aid in factoring out the causes of performance differences. To accomplish this, one often needs artificial domains in which one can systematically vary domain characteristics. In tasks that involve planning and execution, some important characteristics include the complexity of the problem (e.g., the number of obstacles and length of the path in navigation tasks), the reliability of the domain (e.g., the probability that the agent's effectors will have the desired effect), and the rate of environmental change not due to the agent's actions. One can also view resource limitations (e.g., time or energy) as independent variables that affect task difficulty.

Different goals are appropriate for different stages of a developing experimental science. In the beginning, one might be satisfied with qualitative regularities that show one method as better than another under certain conditions, or that show one environmental factor as more devastating to a certain algorithm than another. Later, one would hope for experimental studies to suggest quantitative relations that can actually predict performance on unobserved situations. This should lead ultimately to theoretical analyses that explain such effects at a deeper level, preferably using average-case methods rather than worst-case assumptions.

However, even the earliest qualitative stages of an empirical science can strongly influence the direction of research, identifying promising methods and revealing important roadblocks. Research on planning and intelligent agents is just entering that first stage, but I believe the field will progress rapidly once it has started along the path of careful experimental evaluation.

Informal Notes on the Nature of an Architecture

Paul Rosenbloom
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292

1. It can be very difficult to come up with a firm evaluation of an architecture along particular dimensions. Occasionaly you find clear-cut cases where an architecture achieves optimal performance without requiring anything to be added or modified (either to the architecture itself, or on top of it [such as new knowledge]), or where an architecture is fundamentally incapable of exhibiting a certain form of behavior. However, most of the architectures are flexible enough to do just about anything if you put enough ingenuity into how it is used (this is the "Turing tarpit" problem). In such systems the tough questions come up in the large grey area between the two extremes. This is where questions like the following arise:

   - To what extent is the architecture implicated in the capability?
   - Does it provide the entire capability?
   - Does it provide support that could not be provided any other way?
   - Does it constrain how the capability is manifested?
   - Does the architecture say nothing about the capability?
   - Must an interpreter be built on top of the architecture in order to effectively provide the capability?
   - To what extent is the capability general across application domains?
   - To what extent does the capability integrate with the other requisite capabilities?
   - How much (and what type of) "user" effort is required to get the system to exhibit the capability?

   Without answering such questions, data about which architectures can perform which benchmark tasks at what levels of performance are impossible to interpret. Even once you've answered these questions, there is still the question as to whether it is "appropriate" for the capability to be exhibited as it is – for example, certain types of behavior probably should be performed interpretively. We've written a paper in which we tried to provide preliminary answers to some of these questions for Soar, but it definitely isn't easy.

2. Ideally the set of benchmarks should not overstress any one particular capability over and above the others. What's needed is a set of benchmarks which stress different combinations (and numbers) of the basic capabilities. Of course, the real situation is even more complex that this, because "individual" capabilities – such as learning – really are a whole set of related, but distinct, capabilities. It is difficult (impossible?) to construct a set of benchmarks that doesn't bias the evaluation towards some specific capabilities, or even specific variations of the individual capabilities (such as explanation-based learning versus induction). If a level playing field cannot be achieved, there needs to be a discussion up front about the biases that lead it to slope in one direction or other.

Slides from the Presentations

We felt that presentations on specific tasks and metrics would help focus the discussion, so we invited three people well known for their research and experience with system evaluation: Martha Pollack, Rod Brooks, and Barbara Hayes-Roth. Dr. Pollack was expected to discuss experience gained with a simulated environment; Dr. Brooks was invited to discuss his significant experience with building real robot devices; and Dr. Hayes-Roth was asked to speak about her experience with the application of large knowledge-based systems to practical problems in monitoring and control. We felt that these three individuals represented an interesting range of applications and possible evaluation metrics. In this paper we do not attempt to provide a summary of the talks, but instead simply reproduce the speakers' slides (with permission) on the following pages.

Speakers and Topics

Rod Brooks
Benchmarks and Metrics for Mobile Agents

Martha Pollack
Tunable Benchmarks for Agent Evaluation

Barbara Hayes-Roth
Intelligent Monitoring: Environment, Behavior, Metrics

Benchmarks and Metrics for Mobile Agents

Rod Brooks
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139

Mobile Agents

- Enormous Test Space
- Many Benchmarking Challenges
- Simulations are Extremely Difficult (to do well)
- Strawman Tasks & Metrics
- Standards

Challenges for Benchmarking

What is a task?

- Can't supply a standard canned "sequence" of inputs
- System behavior may be very unstable (within sensor noise) in a formal sense (not necessarily a bad thing)
- Performance may depend critically on sensor suite, actuator suite

Axes of Situation Space

- Indoor/Outdoor/Space Structure
- Static/Dynamic
- Night/Day
- Clear/Rainy-Snowy-Sleety
- World is Engineered/Unengineered
- Map Supplied/Unsupplied

Simulations

- Are doomed to succeed
  (David Miller - who stole it from Takeo Kanade)

- Are generally sensation starved

- Are hard to transition to sensation rich environments

- Usually make you concentrate on the 'wrong' problems

Strawman Tasks

- Navigation - get from A to B
  (how specified?)

- Map building (what representation & why?)

- Report on something

- Retrieve something

- Deposit something

- Remain concealed

Strawman Metrics

- Speed
- Reliability
- Computation Consumption
- Level of Dynamicness handled (??)

Standards

- Standard computational components
- Allowable sensors
- Standard hardware platform

Standard Hardware Platform

- Adequate sensors
- Adequate manouverability
- Low cost (everyone needs one)
- Adequate computation/high speed data line

Tunable Benchmarks for Agent Evaluation

Martha E. Pollack

SRI International

Benchmarks and Metrics Workshop

June 25, 1990

Why *Tunable* ?

Appropriateness of an agent architecture depends on the characteristics of: its environment
its intended tasks

Therefore want to investigate correlations between agent-design choices and environmental factors

(Integration Question)

2

Example: Tileworld

[Pollack & Ringuette, AAAI90]

Simulated, abstract environment with associated tasks

Simulated robot agent

Both highly parameterized

3

The Tileworld Environment

```
# # # # # # # # # # # # # # # # # # # #
#     T   T         T       T         #
#       #                   2 2     T #
#       #                   2         #
#       # # 5             T           #
#     # # # 5 T                       #
#         # 5   T         a         T #
#       T                             #
#         T     T             T T     #
#               # T # T   T           #
#       T       # # # #               #
#     #           # #         T       #
#     #   T   T   T       T           #
#     #                       #       #
#                         T   #   #   #
#     T                   # # # # #   #
#     # # # # #                       #
#     #                     T T       #
#                   T         T     T #
# # # # # # # # # # # # # # # # # # # #
```

a = agent

T = tile

# = obstacle

<digits> = hole

4

Projection of Real-World Environments

- Robot Delivery

  hole appearance = delivery request

  hole = delivery destination

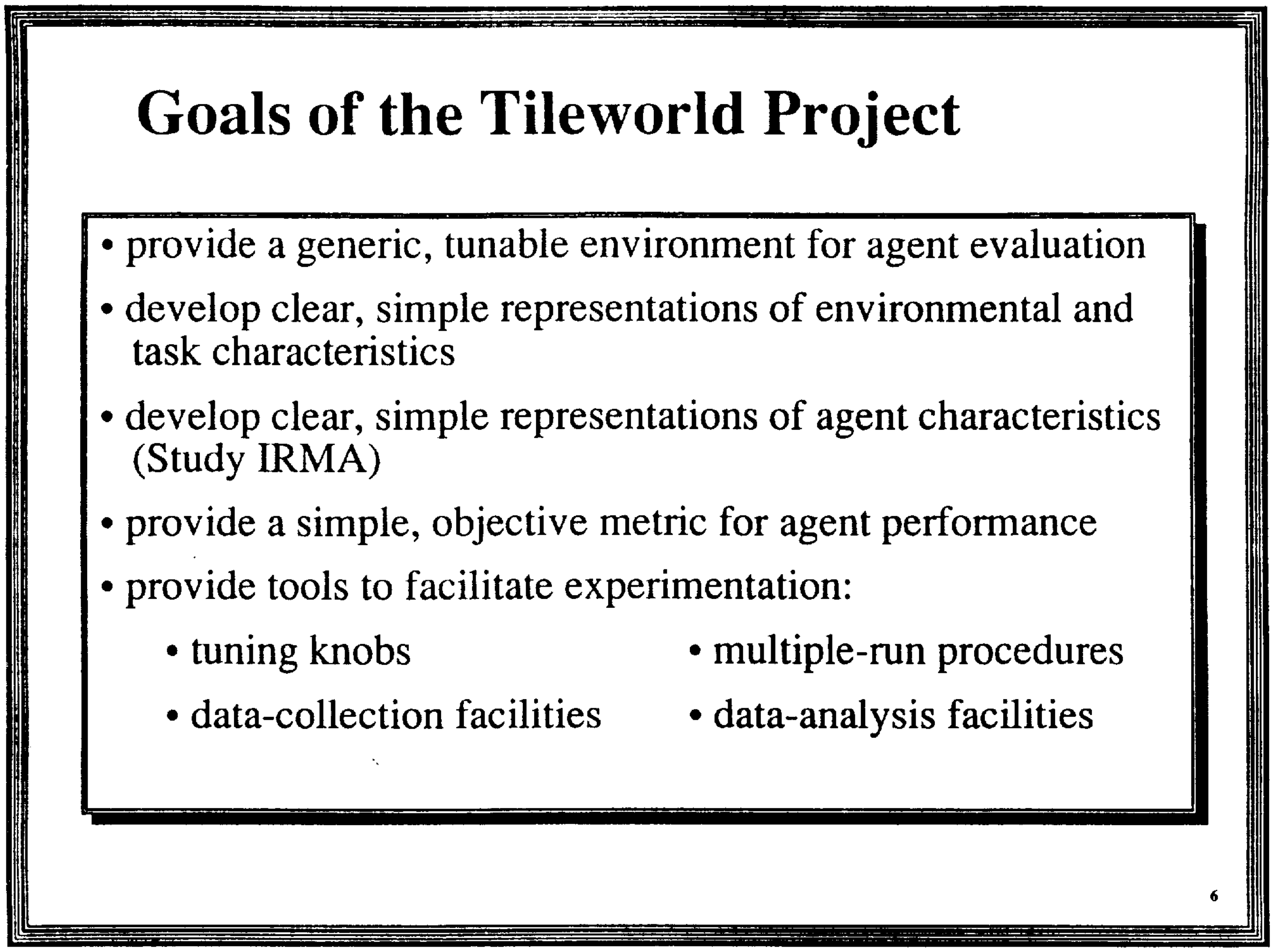
  tile = message or object to be delivered

  agent = robot

  grid = hallways

  simulator time = real time
- Logistics Planning
- Malfunction Handling
- Load Scheduling

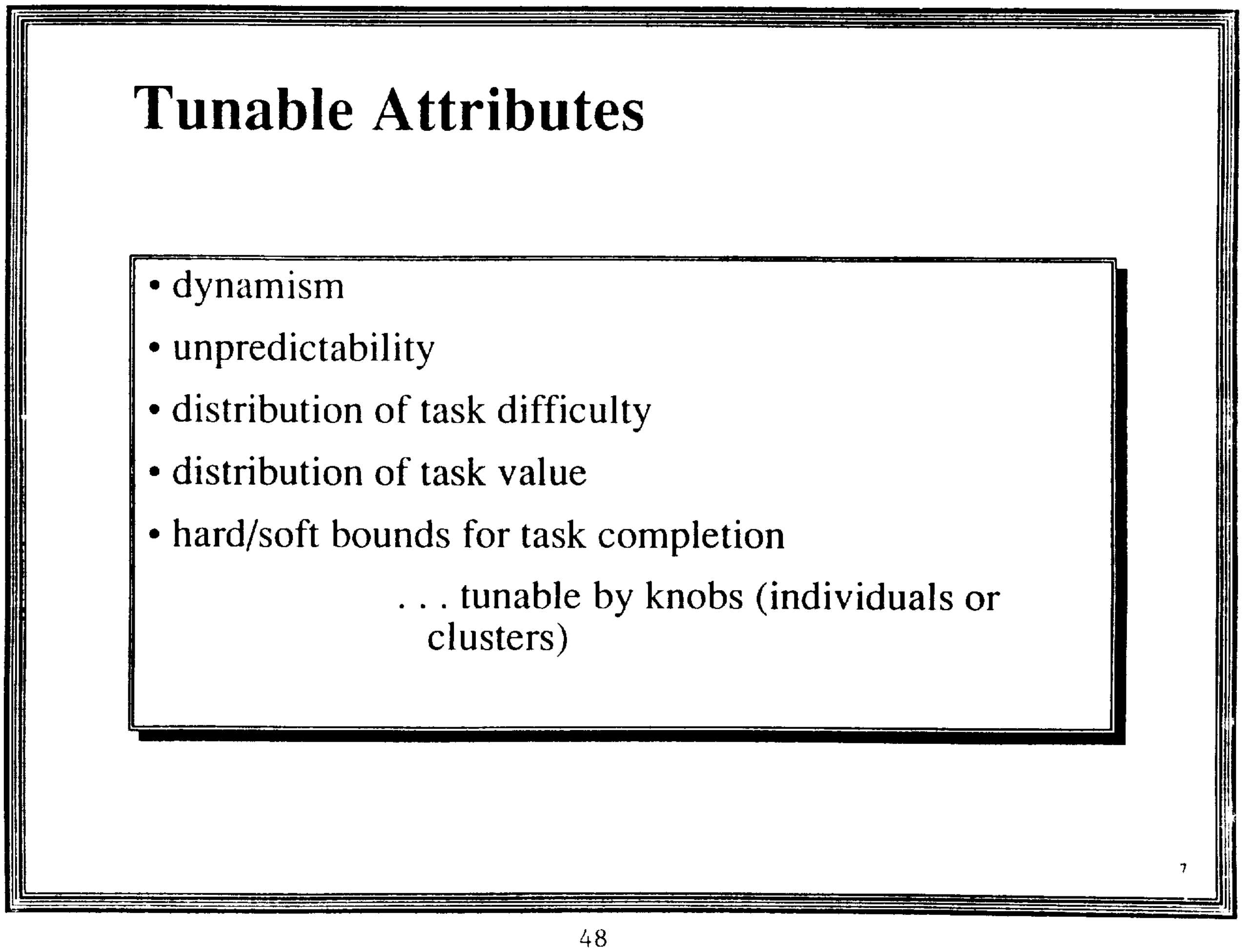
5



Goals of the Tileworld Project

- provide a generic, tunable environment for agent evaluation
- develop clear, simple representations of environmental and task characteristics
- develop clear, simple representations of agent characteristics (Study IRMA)
- provide a simple, objective metric for agent performance
- provide tools to facilitate experimentation:
  - tuning knobs
  - multiple-run procedures
  - data-collection facilities
  - data-analysis facilities

6

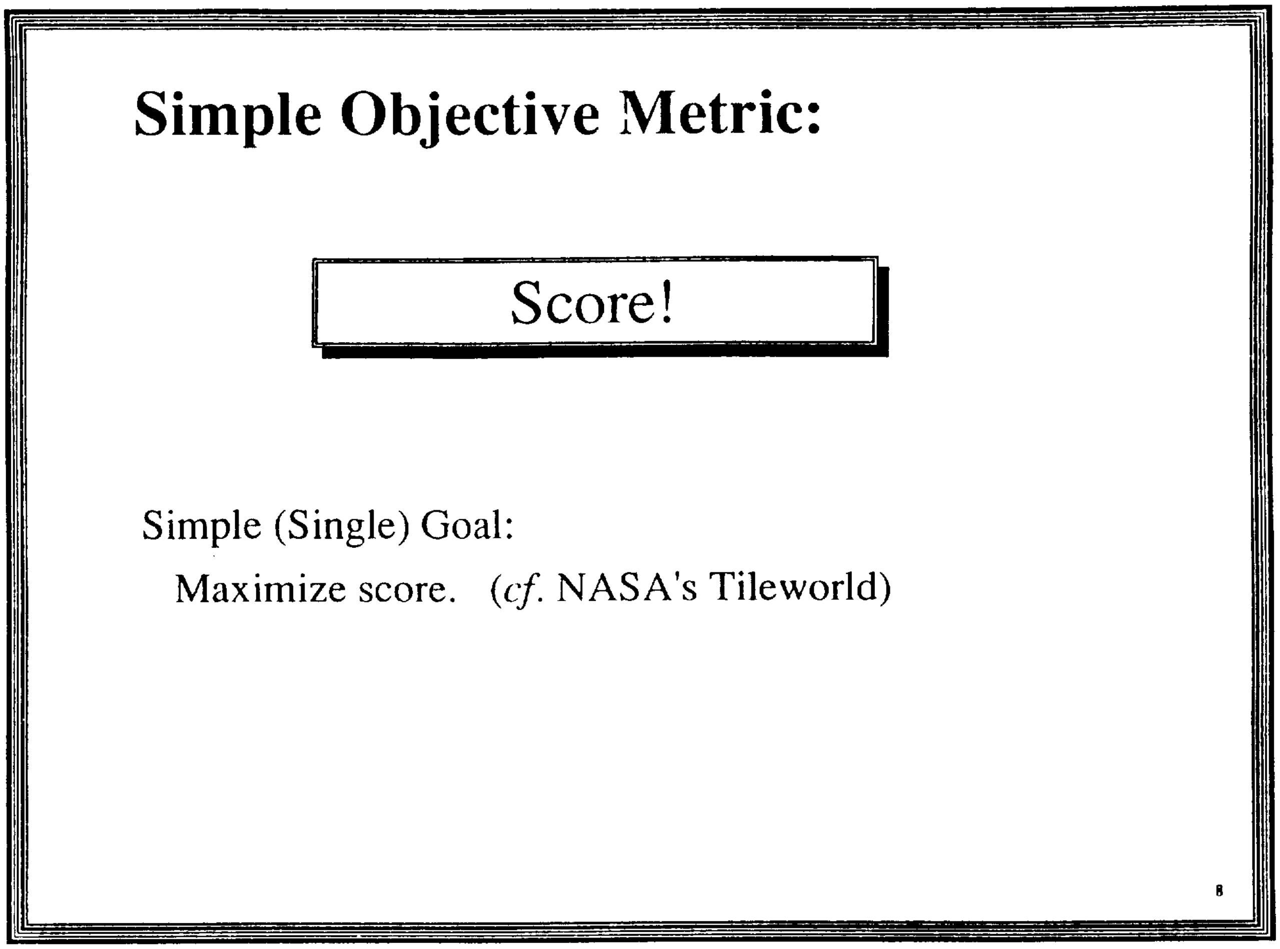


Tunable Attributes

- dynamism
- unpredictability
- distribution of task difficulty
- distribution of task value
- hard/soft bounds for task completion

. . . tunable by knobs (individuals or clusters)

7



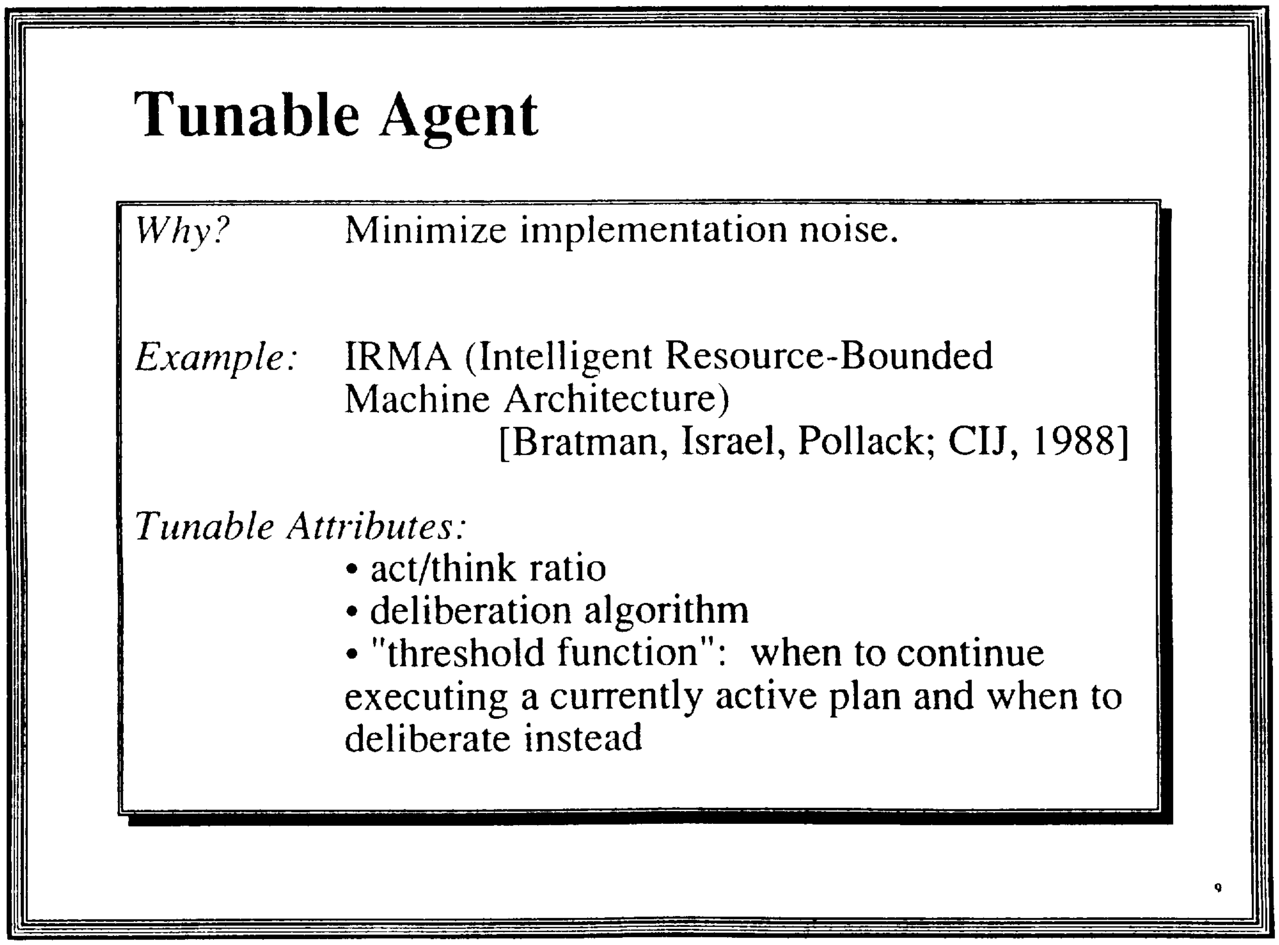
Simple Objective Metric:

Score!

Simple (Single) Goal:

Maximize score. (*cf.* NASA's Tileworld)

8



Tunable Agent

Why?	Minimize implementation noise.
Example:	IRMA (Intelligent Resource-Bounded Machine Architecture) [Bratman, Israel, Pollack; CIJ, 1988]

Tunable Attributes:

- act/think ratio
- deliberation algorithm
- "threshold function": when to continue executing a currently active plan and when to deliberate instead

9

Experimental Analysis

Goal: Specify mapping
from Environment/Task Attributes plus
some Agent Attributes (e.g., act/think ratio)
to Remaining Agent Attributes

10

Related Systems

Phoenix [Cohen et. al., AI Magazine, 1989]
- tightly coupled to firefighting domain
- metrics not well-defined
- complex!

MICE [Durfee & Montegomery, Wkshp. on DAI, 1989]
- focus on inter-agent coordination

NASA's Tileworld
- much richer goal structure
- specification of tunable parameters and metrics more difficult

11

Where next?

- Experimentation !!!
- Environment Extensions:
  - -- e.g., hostile environment
- Agent Extensions:
  - -- e.g., perceptual reliability
- Experimental Framework Extensions
- Adaption of the methodology to other environments, e.g., ones without tight time constraints
- Learning/Self-Tuning

The Moral:

Benchmarks and metrics can be a useful component of the research program in AI planning, but they should be multi-dimensional and tunable.

Intelligent Monitoring: Environment, Behavior, Metrics

Barbara Hayes-Roth
Stanford University

June, 1990

Characteristics of the Environment

Continuous Data

Uncertain/Unreliable Sensor Data

Faulty Sensors

Data Glut

Need for Situation Assessment

Data Distribution

Multiplicity of Conditions

Diversity of Conditions

Real-Time Constraints

Predictability

Potential Interactions

Underlying Model/Knowledge Available

Variable Stress

Characteristics of Behavior

Continuous sensing, computing, acting

Diverse Responses:

- Detection, Diagnosis, Action
- Prediction, Plan, Course of Action
- Interpretation, Explanation, Summarization

Multiple Competing/Complementary Goals

Real-Time Constraints

Resource Bounded

Performance Metrics

Correctness

Timeliness

Responsiveness

Selectivity

Recency

Coherence

Flexibility

Robustness

Scalability

Global Utility

Maintain the overall value of total behavior within an acceptable range over time.

Some Examples:

U1 = Sum (Correctness * Importance)
For All Responses within Deadline

U2 = Sum (Value * Importance),
-1 <= Value <= +1
= F(Correctness, Speed)

U3 = If OK Response to Critical Events
Then (U1 or U2), Else 0

MONITORING AND DIAGNOSIS OF SEMICONDUCTOR MANUFACTURING EQUIPMENT

Dr. Barbara Hayes-Roth
Janet Murdock

Computer Science Department
Stanford University

COLLABORATORS:

Robert Dutton (EE)

Gene Franklin (EE)

P. Khuri-Yakub (EE)

Nils Nilsson (CS)

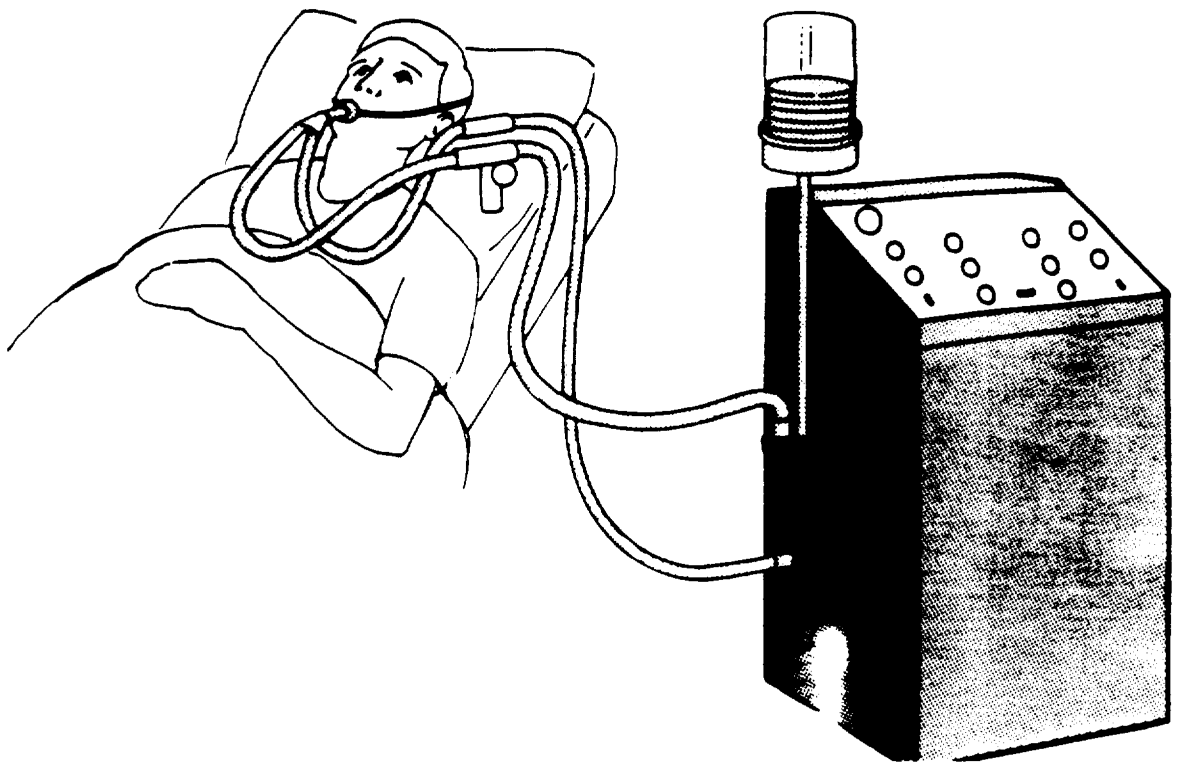
Krishna Saraswat (EE)

Edward Steinmueller (Business)

Gio Wiederhold (CS)

Ernest Wood (EE)

SICU Monitoring



Project Leader: Barbara Hayes-Roth

Students: Rich Washington, David Ash

Post-Docs: Noi Hewett, Anne Collinot,

Visitors: Angel Vina

Collaborator: Dr. Adam Seiver, Palo Alto VAMC

REPORT DOCUMENTATION PAGE

OMB No. 0704-0188

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

[illegible] 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**