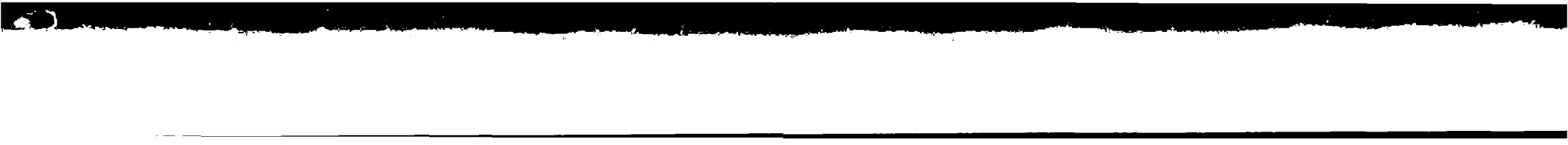
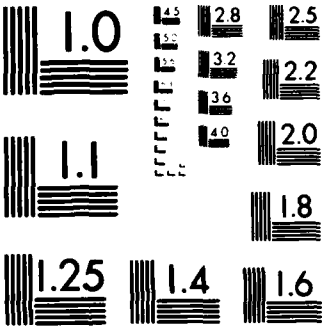
AD-A149 021 PREDICTING THE PERFORMANCE OF DISTRIBUTED KNOWLEDGE-BASED SYSTEMS: A MODELING APPROACH(U) MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND INFORMATION SCIENCE J PAVLIN 1984 F/G 12/1 1/1

UNCLASSIFIED NL





PREDICTING THE PERFORMANCE OF DISTRIBUTED KNOWLEDGE-BASED SYSTEMS: A MODELING APPROACH

Jasmina Pavlin

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts, 01003

ABSTRACT

A model of a distributed knowledge-based system is presented. The model captures the features specific to those systems, such as alternative paths to the solution, utilization of inexact and/or incomplete knowledge and data, dynamic task creation, complex subproblem dependencies and focusing aspect of problem solving. The model is applied to the analysis of communication policies in a distributed interpretation system. The result of the analysis is the best policy for the given environment and system conditions. Another use of the model as a real-time simulation tool is suggested.

I INTRODUCTION

The development and performance-tuning of a knowledge-based system like HEARSAY-II [2] is still mostly an art. Ideally, one would like to have a set of *equations which relate system's input, internal structure* and output, as in classical control theory. This would allow complete analysis of the system and predict its behavior for any input. Unfortunately, in such complex systems the interaction of many parameters precludes such characterization. We feel that even with a complex system a limited analysis should be attempted, and that this is possible with an appropriate modeling procedure.

One of the first attempts at modeling a knowledge-based system was done by Fox [1], but his model is too abstract to deal with the phenomenon of subproblem interaction, an important factor in system performance. His model is also limited in applicability because it is a static approximation, namely, time relationships among processing elements are not considered.

In our earlier work on system measures [3], we addressed one aspect of the performance-tuning problem: *what would be the change in system performance if a component with different characteristics was introduced?* Due to the nature of the question, that work concentrated on the model of a component (knowledge source, scheduler), and it relied on system mechanisms for component interaction.

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8006327 and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

This work centers around a different question: what would be the change in performance if a different relationship among the components was introduced? We develop a model of the complete system and its environment, in which a processing component is relatively simple, and the focus is on the interaction among the components. Since our intention is to model a distributed system, both the interaction among knowledge sources and the interaction among the nodes are considered.

In the following sections, we describe the model of a distributed knowledge-based system (DKBS), show an example if its application as an analysis tool, and suggest its application as a real-time simulation tool for these systems.

II THE MODEL

There are a number of features that need to be incorporated in any realistic performance model of a DKBS:

1. The system usually works on a single problem, which is divided into many subproblems with complex dependencies, so that allocation of one subproblem can not be considered independent of the others.
2. The solution is derived by employing a limited search, in which there is no full enumeration, but only promising alternatives are explored. Thus, many tasks are created during processing, they are not known before the processing starts.
3. Processing in DKBSs is often characterized by uncertainty, since input data may be inaccurate or missing. Also, the problem on which the system is working is often so complex that the solution methods have been only partially identified.
4. In order to reduce the uncertainty in the problem, there is often redundancy in the solution process. It originates either in alternative views of the environment, or in different types of knowledge applied to the same data. Alternative solution paths are formed, and there are many possible tasks involved in the solution of the problem. The focusing problem becomes a crucial aspect of successful processing, and the main issue centers not around the question who will do the work, but if anybody needs to do it at all.

We build a model starting with a Petri-net representation [5]. The Petri-net formalism has the basic concepts necessary for modeling a distributed system with asynchronous processing: the notion of events which occur under certain conditions and the ability to represent the asynchronous nature of events. The basic concepts in the model are activites, domains and data units.

The correspondence between a Petri-net and a DKBS is shown in Table 1. An *activity* represents a knowledge application process, while a *domain* is a part of the environment or the processing space from which the activity takes its input. Data contained in a domain we call a *data unit*. In a HEARSAY-like system, for example, an activity would be a group of knowledge source invocations, a domain would be a part of the area of interest, and a data unit would be a group of hypotheses. Domain boundaries and the scope of an activity are both application dependent. Their choice for a particular application will be shown in the example section.

The system configuration is defined as a fourtuple:

N=(D,A,I,O)
D-the set of domains
A-the set of activities
I:A to D^*- input function
O:A to D^*- output function

Input and output functions specify input and output domains for an activity.

A state of the network is defined by the placement of data units:

$P=(n_1, \ldots n_s)$
s-number of domains in D

where n_i is a number of data units at domain i in D. The system executes by changing its state. The state is changed by performing an activity, which causes data units to be created at its output domains. An activity can be performed only if it is enabled, that is, if it has a data unit in each input domain. The execution ends when a data unit is created in one of the system output domains (those which are not input to any activity).

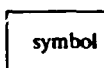
symbol	Petri-net	interpretation system
	transition	activity
	place	domain
•	token	data unit

Table 1: The correspondence between a Petri-net and a DKBS system.

Interpreted in this way, Petri-nets become a convenient formalism for depicting static relationships among the components in a DKBS, but it lacks a dynamic system characterization. The system is more successful if its solution is more accurate, or achieved in a shorter time; a succession of activities which leads to such a solution represents a good allocation strategy. Accuracy and time are then essential characteristics of both activites and data. We depart from basic Petri-nets by augmenting data units with attributes and activities with transition functions. Also, in order to capture the focusing aspect of problem solving, we define execution rules, specifying which among the possible tasks will be performed.

The time required to perform an activity is a function of the amount of work that needs to be done. We have chosen the concept of volume as the simplest estimate of that amount of work. Thus, we define a data unit as a triple:

d=(v,a,t)
v-the volume of data
a-the accuracy of data
t-the time of arrival of data.

The value of each of these attributes in the model is an estimate that needs to be obtained from the real system by some sampling process. In a HEARSAY-like system, for example, the volume is an estimate of the number of hypotheses, the accuracy is their belief, and an estimate of the arrival of data is the time atribute.

Let us denote ds a data unit which represents the solution of the system:

ds=(vs,as,ts).

In general, a higher accuracy will be achieved by combining more independent views on the problem, at the expense of longer solution time. Consequently, the objective of the system represents a trade-off between these two opposing requirements, and we define the performance evaluation function to be the ratio of the accuracy and the elapsed time:

J=as/ts.

An activity is seen as performing three functions: fv, fa and ft on the attributes of input data. Let input data units be denoted:

$d_i=(v_i,a_i,t_i) \qquad i=1,\ldots,n$

and the output data unit:

d=(v,a,t).

Then the functions of an activity can be represented as follows:

$v=fv(v_1,\ldots,v_n)$
$a=fa(a_1,\ldots,a_n)$
$t=ft(v_1,\ldots,v_n, t_1,\ldots,t_n)$.

The fv, volume transition function, determines the volume of the output based on the volumes of inputs.

The fa, accuracy transition function, determines the accuracy of the output based on the accuracies of inputs.

The ft, or time transition function, determines the time of creation of output data. The output time depends both on input times and the volume of data.

An activity is performed if it has the highest priority among enabled activities (those which have data in their input domains). The *execution rule* specifies this priority relationship.

A critical factor in the model's applicability is determination of the transition functions used by the model. They may be hard to determine accurately, especially if the intention is to use the model in the design phase, when a working system is not available. However, the functions can be stated in rather general form (as will be shown in the example) and fine tuned in the verification phase.

III EXAMPLE APPLICATION

As an example application, let us consider the use of the model in determining appropriate communication strategies for the distributed vehicle monitoring testbed [4]. The testbed simulates a distributed interpretation system whose goal is to create a dynamic map of vehicles moving through the system's environment. Vehicles emit acoustic signals which are identified and roughly located by sensors. Sensors report this information to nearby nodes. Every node is an architecturally complete HEARSAY-II system.

In order to create a map, every vehicle or a formation of vehicles (pattern) has to be identified, located and tracked. A vehicle is identified by a number of groups, corresponding to its different acoustic sources (engine, fan). Groups correspond to signals related by the same harmonic frequency. Thus, four levels of abstraction can be identified in the solution process: signal, group, vehicle and pattern. For this example, we assume that signal tracks are formed first, and then combined into tracks on higher abstraction levels.

Let us consider the system with two nodes which partially overlap in their input domains. For this example, we consider only single vehicle formations moving in one direction. The nodes are positioned along that direction, so that node 1 receives input data first. The solution is to be formed at node 2. It is then appropriate that node 1 should send information to node 2. We want, with the help of the model, to answer the following questions:

1. What type of information should be communicated: exclusive (non-overlapping), shared (overlapping), or all (overlapping and non-overlapping)?
2. Should the information be communicated on a low level of abstraction (group) or on a high level (pattern)?

Six possible configurations, coresponding to different combinations of communication level and communicated information, will be examined:

a. Communication of non-overlapping information, on a low level.

b. Communication of non-overlapping information, on a high level.

c. Communication of all information on a low level

d. Communication of non-overlapping information on a low level, overlapping information on a high level.

e. Communication of overlapping information on a low level, overlapping information on a high level.

f. Communication of all information on a high level.

For this problem we define four types of activities:

1. Synthesis (S) whose results are data on higher abstraction level.
2. Merging (M) whose results are data of a larger scope (longer tracks).
3. Unification (U) which combines different views of the same events.
4. Communication (C) which moves data from one node to the other.

The transition *functions are based on the observations* of the testbed behavior. Their definition is summarized in Table 2. The execution rule used in the simulation assumes the following priority relation when more than one activity is enabled:

$$C > U > M > S$$

activity	fv	fa	ft
synthesis	$\Sigma(Vi)$	$cs^{*}AVG(Ai)$	$MAX(Ti)+tp^{*}\Sigma(Vi)$
merging	$\Sigma(Vi)$	$cm^{*}AVG(Ai)$	$MAX(Ti)+tp^{*}\Sigma(Vi)$
unification	Vi	MAX(Ai)	MAX(Ti)
communication	Vi	Ai	Ti+tc

Table 2: Definition of transition functions.

fv-volume transition function
fa-accuracy transition function
ft-time transition function
di-input data unit, di=(Vi,Ai,Ti)
Cs,Cm-knowledge power constants
Cs Cm=1
tp-time to process unit volume, tp=1
tc-time to communicate unit volume, tc=1

We define four input data units:

1. d11 is the input from the domain exclusive to node 1
2. d12 is the input that node 1 collects from the overlapping domain.
3. d21 is the input that node 2 collects from the overlapping domain.
4. d22 is the input from the domain exclusive to node 2.

The model is simulated for the following data definition

d11 =(2, 0.6, 0)
d12 =(3, 0.6, 0)
d21=(3, 0.4, 4)
d22=(2, 0.4, 4)

Figures 1 to 6 show all the activities performed in each configuration with a given execution rule, before the solution is reached. The input domains are marked by

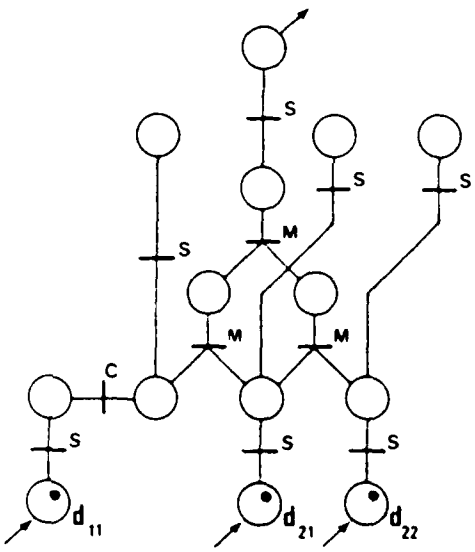


Figure 1: Configuration a.

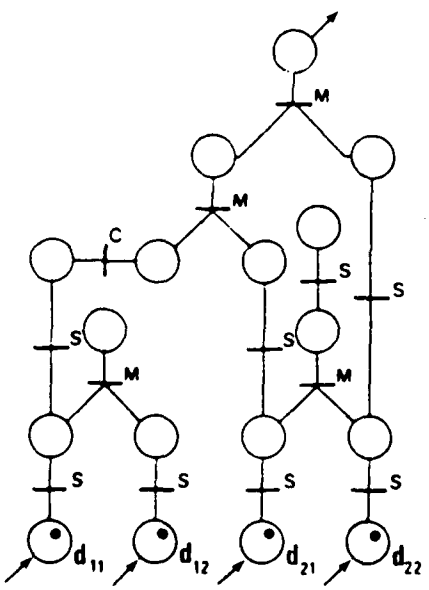


Figure 2: Configuration b.

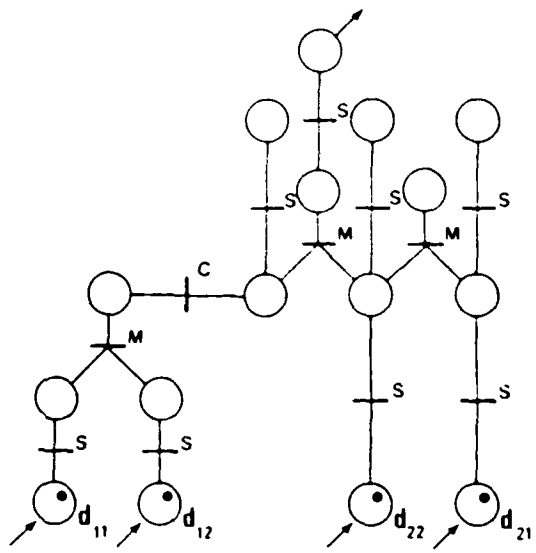


Figure 3: Configuration c.

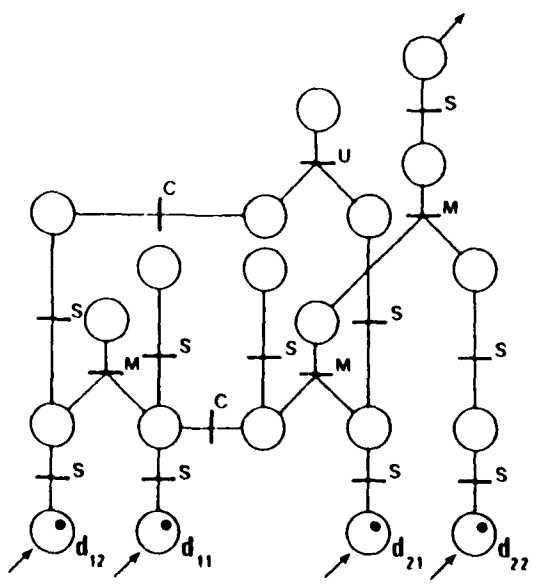


Figure 4: Configuration d.

incoming arrows, the output domain is marked by an outgoing arrow. Performance is judged by comparing the values of the objective function, J, for different configurations

The values of the solution accuracy (on zero to one scale),solution time (in number of system cycles), and the objective function obtained by model simulation and coresponding experiments in the testbed are shown in Figures 7 to 9. Both the simulation and the experiments show configuration f to be the best Furthermore, the ordering of configurations is preserved in the simulation results, serving as a limited verification of the model.

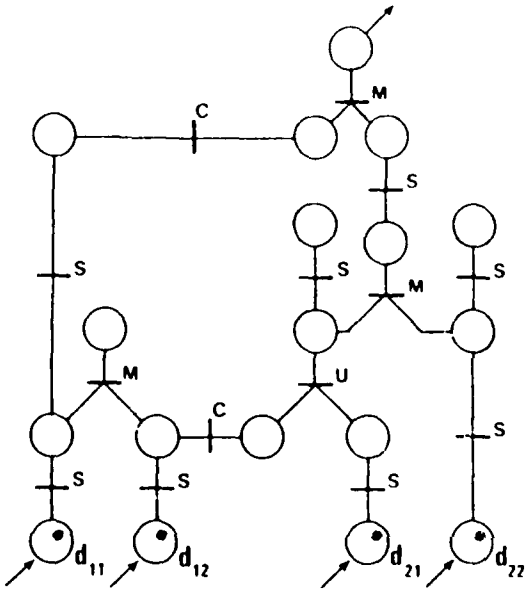


Figure 5: Configuration e.

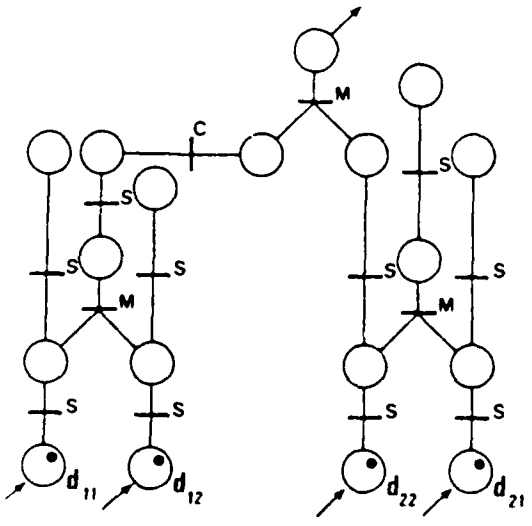


Figure 6: Configuration f.

IV DISCUSSION

An important step towards the control of complex systems is the analysis of the relation between the *environment*, *system structure* and the *performance* We have devised an approach in which a limited analysis of one aspect of that problem is possible: finding a best class of communication policies for a distributed interpretation system operating in a simplified environment.

The approach may prove useful outside simple and structured environments amenable to analysis. We believe, (based on initial results) that the policies are relatively robust to limited changes in environmental conditions, so that the results also hold for more realistic environments similar to the analyzed ones. Also, the results of the analysis can be transfered to more complex environments, if

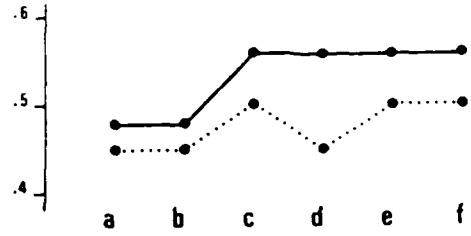


Figure 7: Solution accuracy.

—— testbed ····· model

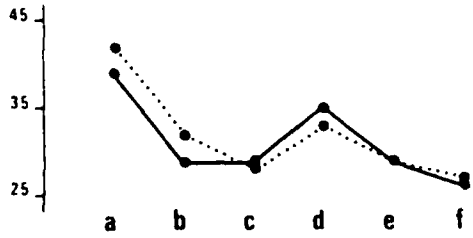


Figure 8: Solution time.

—— testbed ····· model

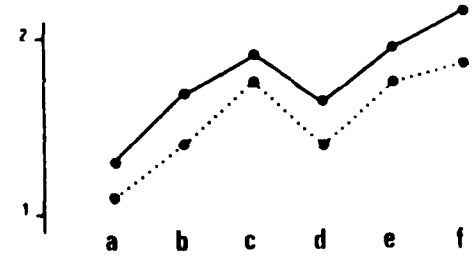


Figure 9: Objective function.

—— testbed ····· model

they can be treated as a combination of the simple environments

Another use of the model is as a system stand-in. A system supplied with the results of the analysis has a stored communication policy for an analyzed environment; a system supplied with the model and confronted with a novel environment can simulate a number of communication structures. Although without the analysis benefits of complete search and a global optimum, this may be a very useful guide in the choice of communication policies.

ACKNOWLEDGMENTS

I am grateful to Daniel D. Corkill and Victor R. Lesser whose careful reading and helpful suggestions led to an improved version of the original manuscript.

REFERENCES

1. Mark S.Fox
"Organizational Structuring: Designing Large Complex Software."
Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburg, Pennsylvania, December 1979.

2. Victor R. Lesser and Lee D. Erman
"A Retrospective View of the Hearsay-II Architecture."
In *Proc. IJCAI-77*, pp. 790-800.

3. Victor R.Lesser, Scott Reed and Jasmina Pavlin
"Quantifying and Simulating the Behavior of Knowledge-based Interpretation Systems."
In *Proc. IJCAI-1979*, pp. 111-113.

4. Victor Lesser, Daniel Corkill, Jasmina Pavlin, Larry Lefkowitz, Eva Hudlicka, Richard Brooks, and Scott Reed
"A high-level simulation testbed for cooperative distributed problem solving."
Proceedings of the Third International Conference on Distributed Computer Systems, pages 341-349, October 1982.

5. James L. Peterson
"Petri Net Theory And Modeling of systems."
Prentice-Hall Inc., Englewood Cliffs, N.J. 07632, 1981.

END

FILMED

2-85

DTIC