A Two-layer Learning System for Document Retrieval in Multiple-User Environments

by

U. Mukhopadhyay, M. N. Huhns, R. D. Bonnell and L. M. Stephens
Center for Machine Intelligence
University of South Carolina

ABSTRACT

MINDS (Multiple Intelligent Node Document Servers) is a distributed system of knowledge-based query engines for efficiently retrieving multimedia documents in an office environment of distributed workstations. By learning document distribution patterns, as well as user interests and preferences during system usage, it customizes document retrievals for each user. A two-layer learning system has been implemented for MINDS. The knowledge base used by the query engine is learned at the lower level with the help of heuristics for assigning credit and recommending adjustments; these heuristics are incrementally refined at the upper level.

I. INTRODUCTION

Documents are used in computerized office environments to store a variety of information. This information is often difficult to utilize, especially in large offices with distributed workstations, because users do not have perfect knowledge of the documents in the system or of the organization for their storage. The goal of the MINDS project is to develop a distributed system of intelligent servers that 1) learn dynamically about document storage patterns throughout the system, and 2) learn interests and preferences of users so that searches are efficient and produce relevant documents [1,2]. The strategy adopted for evaluating a set of learning heuristics that are applicable to this goal is presented. In particular, this paper describes the heuristic evaluation testbed, distance measures for metaknowledge, document migration heuristics, evidence assimilation techniques, and results of a system simulation.

II. DISTRIBUTED WORKSTATION ENVIRONMENT

A. Organization of Documents

Queries regarding documents are frequently content-based. Automatic text-understanding systems could conceivably process these queries but would be expensive to develop and use. Document names are not descriptive enough and can not be relied on exclusively for processing content-based queries. However, a set of keywords may be used to describe document contents: the retrieval of documents can then be predicated on these keywords

---

This research was supported in part by NCR Corporation.

as well as other document attributes, such as author, creation date, and location. Complex qualifiers, which are conjunctions or disjunctions of predicates on these attributes, may also be used. Each document is represented by a surrogate containing its attributes. The document and its surrogate are subsequently updated or deleted as dictated by system usage. Surrogates occupy only a fraction of the storage space required by the documents, but usually contain enough information for users to determine whether a document is useful.

Each workstation supports a single user who may query the system. Documents owned by this user are stored locally but may be accessed by users at other workstations. Though multiple copies of documents may be stored in the system, a user may not have the same name for two of his documents unless they have different version numbers. Document migration within the system is allowed and is viewed as a combination of copying a document by a user and then deleting it from the document base of the original owner.

Documents are logically organized in a file-cabinet paradigm. Users own cabinets, each of which contains drawers, which in turn contain folders with sub-folders and documents in a hierarchy. The classification of documents in this manner enables users to narrow down the search space in many instances and is especially conducive to browsing through documents.

B. The User's Perspective

In typical distributed document management systems, directories are either centralized or distributed, with or without redundancy [3]. However, the directory information is consistent throughout the system; information is stored redundantly only to reduce directory access time. The algorithm for document retrieval consists of matching predicates for retrieval with the document properties stored in the directory. The documents for which the match is successful are then retrieved from the indicated storage addresses. Since the directory information is consistent throughout the system, the response to a query is the same without regard to the identity of the query originator.

In a large system, the response to a query may consist of many documents, only a few of which may be relevant to that particular user. Also, the set of documents relevant to a second user may be quite different from that of the first, even though their queries are identical. The problem appears to originate from a lack of specificity in formulating the query. A judicious choice of predicates would apparently cause all the documents that are irrelevant to the query originator to be rejected. However, this would require a sophisticated query language, rich enough to allow the expression of a user's short-term and long-term goals, plans, and interests. A comprehensive framework for document surrogates, possibly along the lines of semantic nets, would also be required. Formulating queries would be extremely cumbersome and the increased power of the system would be offset by the additional effort demanded from the user.

In the absence of any information about the user, whether explicitly stated in the query or embedded in the system knowledge base, the response will necessarily consist of a superset of the sets of relevant documents described by the query from the perspective of each user. User-transparency in a large multiple-user environment may thus cause a query response to contain an arbitrarily large number of irrelevant documents. It is our view that systems of the future need to maintain models of their users in a background mode in order to make document searches more efficient and productive without burdening the user.

MINDS is a distributed document server with some special characteristics that allow personalized document retrieval. Additional information, in the form of personalized document metaknowledge, is stored at each workstation to allow the system to scan the document bases of all system users in a best-first fashion from the viewpoint of the query originator. Because MINDS maintains (at each workstation) models of both the current system state and the local user's document preferences, it is analogous to a Class C natural language interface [4].

III. QUERY PROCESSING IN MINDS

A. Workstation Interactions

The MINDS system shares tasks, knowledge, and metaknowledge for cooperation among the workstations. A complex query is processed by first decomposing it into simpler subtasks, with the help of locally stored metaknowledge, such that the search space for a subtask is limited to the documents owned by a system user. Subtasks are then transmitted to the respective workstations where they are processed locally. Responses to the subqueries are transmitted back to the workstation that initiated the subqueries, where the results are synthesized and ranked in decreasing order of relevance as estimated by the metaknowledge. If the subquery is content-based, relevant metaknowledge is also sent to the query originator along with the documents and surrogates constituting the response to the subquery. The transmitted metaknowledge may be used for updating the metaknowledge of the receiver in accordance with the learning strategy. Activities, such as creating, deleting, and copying documents, are also performed at each workstation; some of these functions require cooperation among the workstations. Again, metaknowledge may be modified as a side-effect of these activities.

B. Metaknowledge

Each metaknowledge element (Figure 1) is a four-tuple with fields for two users, a keyword, and a certainty factor [0,1]. The metaknowledge element, (Smith, Jones, compiler, 0.8), represents the following:

> "Smith's past experience suggests that the possibility of finding relevant documents on compilers among those owned by Jones is high (8 on a scale of 10)."

The certainty factor [5] provides an ordering of the search for documents. It reflects 1) the breadth of information at a workstation pertaining to a specific keyword, 2) how useful documents concerning this keyword have proven to be in the past, and 3) how recently the workstation has acquired its information. The metaknowledge is first initialized with default values of certainty factors. A set of heuristic learning rules defines the constraints for modifying these values during system usage.

If a user had metaknowledge for each document rather than for each user of the system, the knowledge would be precise. However, the disadvantages of this approach are 1) for an average of n documents per user, the metaknowledge overhead would be n times as much, and 2) for new documents, no prediction of relevance can be made. On the other hand, there is a positive correlation among the cluster of documents owned by a particular user, so that conceptual-level properties may be assigned to this cluster. Future additions to this cluster would probably have properties similar to the cluster prototype. For example, the metaknowledge stored by Smith associating the document base of Jones with the keyword compiler is a conceptual-level property of Jones' document base. This property can be exploited by Smith to facilitate document searches and is therefore stored at Smith's workstation, rather than at Jones'.

IV. THE LEARNING TESTBED

A. The Learning Cycle

MINDS is being developed for operation in a wide range of office environments. The state of an environment at any instant of time is given by the content and configuration of the metaknowledge and document bases of the system. Commands issued by users comprise the system inputs, and retrieved documents and surrogates constitute the outputs. The state of the system may change as a result of executing a command [6]. The system dynamics are shown in Figure 2.

A learning testbed has been implemented to develop a robust body of learning heuristics applicable to any office environment. In an operational MINDS system, these heuristics are static in that all refinements are made off-line, primarily because of the large amount of computation involved. On-line refinement would also require the identification of a body of meta-heuristics for guiding the refinement process; these are being studied separately.

The heuristics refinement strategy, outlined in Figure 3, is to compare the performance of different versions of MINDS, each implanted with a unique set of learning heuristics. A number of simulations are run with an initial set of heuristics; the results are evaluated and used by the refinement mechanism to create a new set, possibly overlapping the previous one. The new set of heuristics is used for the next simulation. During a refinement cycle, heuristics may be discarded, added, or modified based on their influence in shaping the metaknowledge. Good heuristic refinement rules (meta-heuristics) expedite the search for an optimal set.

Each simulation is run on a data set representing a specific office environment; the data set consists of an initial database of documents and their locations, the initial metaknowledge of the users, and a command sequence representing plausible document transactions. Measurements, which are made periodically during the course of a simulation, are used to evaluate the performance of a given set of heuristics. The same set is then tested on other simulated office environments. Based on the evaluations of each simulation, the heuristics are refined. The modified heuristics are then evaluated on the same simulated office environments.

B. Domain Modeling and Knowledge Representation

The practical value of the heuristics developed in the testbed depends on the validity of the office models used in the simulations. An office is described by aggregate descriptors such as the number of users and the relative frequencies of certain commands. These descriptors are used to generate a distributed document base, metaknowledge for each user, and a command sequence.

Although document retrievals may be based on several types of predicates such as authorship and location, only content-based (keyword) retrievals are considered in the simulations since the other types of retrievals do not modify the metaknowledge. Commands which do not affect the system state, and thus are not important for learning, are not included.

In an operational MINDS system, documents and surrogates would be stored in separate data structures at the same workstation. When a document is moved, its surrogate would also be moved. Surrogates suffice for the processing of retrievals, since these commands return only the names and locations of documents that satisfy the search predicates; actual documents, however, would have to be used for processing reads. After a document is read by a user, he would be asked to provide a relevance factor for the document on a [0,1] scale. Metaknowledge updates are influenced by this value.

In the simulations, frames are employed for capturing a combined view of the surrogate and the document. Each document has a name and several keywords describing it. All other attributes in the surrogate model are discarded since all retrievals in the simulations are keyword-based. A read operation with a content-based predicate is a keyword-based retrieval that culminates in the reading of one or more documents from the top of the list of retrieved documents. In an actual system, a user would then provide a relevance factor for each document read. If the documents were to be retrieved and read by another user, the relevance factors would probably be different. Also, if the same user evaluated the same documents in terms of some other keyword, the relevance factors would probably be different. In the simulations, the contents of each document are not stored, only the relevance factor it would be accorded by each user in terms of each keyword describing it.

If documents are distributed uniformly such that there is no preferred sequence of workstations to search, the metaknowledge will not prove helpful. However, instead of employing an exhaustive search strategy, people in offices (computerized or otherwise) always seem to rely on past experiences to order their searches in a best-first fashion. This suggests that the distribution of knowledge in offices is not uniform.

The correlation among the documents owned by a particular user is modeled in the testbed by biasing the relevance factors associated with the documents. For example, if Jones' documents on compilers are biased from Smith's viewpoint by 0.2, then the relevance factors associating Smith with compilers in documents owned by Jones will have a uniform distribution between 0.2 and 1.0. A bias of -0.4 would cause a distribution between 0 and 0.6. The bias is mutual in that

BIAS(Smith, Jones, compiler) = BIAS(Jones, Smith, compiler).

A typical document base is shown in Figure 4.

Metaknowledge is also stored in frames as shown in the example of Figure 5. Each user has metaknowledge which captures his personal view of the dispersion of relevant documents and consists of certainty factor assignments for all combinations of users and keywords, including his view of the documents owned by himself. A system of n users and m keywords would result in nm certainty factor assignments in each of the n metaknowledge sets. Two choices were considered for initializing the metaknowledge at the start of the experiment. The first, an unbiased assignment of certainty factors (say 0.5), would initially result in ties for determining the best locations to search; if the conflict-resolution strategy is to choose the first location to appear in the list, then the system would tend to learn about users placed at the top of the list earlier than those placed near the end. The second initialization strategy is to randomly allocate certainty factors, possibly with a uniform distribution, to ensure that the learning progresses in an unbiased fashion.

The second strategy was adopted for the simulations reported in this paper.

C. Metaknowledge Update Heuristics

The metaknowledge updating heuristics are based on the paradigm of the intelligent office-worker who conducts an ordered search for documents based on past experiences in the office environment. When Smith asks Jones for documents on compilers and Jones provides one or more documents that are relevant to him, Smith learns that Jones' document base is likely to continue having useful documents on compilers in the future. If Jones has no documents on compilers, or if none of the documents on compilers are relevant to him, Smith will learn that this may not be a good place to search for documents on compilers in the future. In either case, Jones may assume that Smith will continue searching other locations and acquire documents on compilers from some other user. Consequently, Jones will increase his belief in Smith's ability to provide documents on compilers in the future. This increase in belief will probably not be large since Smith's newly acquired knowledge on compilers may not be of the type relevant to Jones.

1. Initial Set of Heuristics

Move and copy commands do not explicitly appear in the command sequence for this set of simulations. They are, however, in some of the learning heuristics for document migration. The results reported here were generated by the following set of heuristics:

Heuristic 1. (also applicable for the DELETE part of MOVE)
IF a document is deleted
THEN no metaknowledge is changed

Heuristic 2.
IF a document is created by user1
THEN metaknowledge of user1 about user1 regarding each keyword of the document is increased to 1.0 (the maximum relevance).

Heuristic 3.
IF a retrieve command predicated on keyword1 is issued by user1
AND at least one user2 surrogate contains keyword1
THEN (a) user1 metaknowledge about user2 regarding keyword1 is increased (weight 0.1)
(b) user2 metaknowledge about user1 regarding keyword1 is increased (weight 0.1)

```
Heuristic 4.
    IF   a retrieve command predicated on keyword1
         is issued by user1
    AND  no user2 surrogate contains keyword1
    THEN (a) user1 metaknowledge about user2 regarding
         keyword1 is decreased to zero
         (b) user2 metaknowledge about user1 regarding
         keyword1 is increased (weight 0.1)


Heuristic 5.
    IF   a read command predicated on keyword1 is
         issued by user1
    AND  no user2 document contains keyword1
    THEN (a) user1 metaknowledge about user2 regarding
         keyword1 is decreased to zero
         (b) user2 metaknowledge about user1 regarding
         keyword1 is increased (weight 0.1)


Heuristic 6.
    IF   a read command predicated on keyword1 is
         issued by user1
    AND  at least one user2 document contains keyword1
    THEN (a) user1 metaknowledge about user2 regarding
         keyword1 is changed, based on the highest relevance
         of all user2 documents regarding keyword1.
         (b) user2 metaknowledge about user1 regarding
         keyword1 is increased (weight 0.1)


Heuristic 7.
    IF   a read command predicated on keyword1 is
         issued by user1
    AND  document1 owned by user2 contains keyword1
    AND  the relevance of document1 to user1 by way
         of keyword1 exceeds the move_copy threshold
    AND  user1 does not have document1
    THEN (a) user1 copies document1 from user2
         (b) metaknowledge of user1 about user1 regarding
         keyword1 of the document is increased to 1.0


Heuristic 8.
    IF   a read command predicated on keyword1 is
         issued by user1
    AND  user1 has copied document1 from user2
    AND  the maximum relevance of document1 to
         user2 by way of any keyword is less
         than the delete_threshold
    THEN document1 is deleted from the document
         base of user2
```

These heuristics have also been written in first-order logic and as OPS5 rules for implementation purposes.

2. Assimilation of Evidence

The learning heuristics shown above enable the metaknowledge to be changed on the basis of new evidence which typically consists of the relevance rating of a document, the observation of a document being copied, etc. The metaknowledge updating scheme should be able to take into account

a. temporal precedence -- the system is dynamic and therefore recently acquired evidence is more indicative of the current state of the system than evidence acquired earlier. If f1 is the mapping function for a downward revision of the certainty factor (contradiction) and f2 is the mapping function for an upward revision (confirmation), then $f2(f1(x)) \geq f1(f2(x))$, for all $0 \leq x \leq 1$ (Figure 6).

b. reliability of evidence -- some types of evidence are more reliable than others. If a surrogate with a desired keyword is successfully retrieved by Jones from Smith, this action by itself does not completely support the proposition that Smith's documents on compilers are going to prove relevant to Jones in the future, since the relevance of this document to Jones is not known. However, if this document is read by Jones, then the relevance value assigned by him constitutes reliable evidence. The reliability of the source is also important for evaluating the metaknowledge sent by a user. When Smith offers metaknowledge to Jones about documents on compilers, Jones will pay heed to it only if he has found Smith to be a reliable source of documents on compilers in the past.

c. degree of support -- the degree of support for a proposition may vary. When a user is asked if the document he has read is relevant to him, his answer does not have to be limited to "yes" or "no" but may be a value in the range [0,1].

d. saturation characteristics -- when the initial certainty factor for a metaknowledge element is high, additional confirmatory evidence will not change (increase) it substantially. However, if the evidence were to be contradictory, the change (decrease) in confidence factor would be high under the same initial condition. The situation is exactly reversed when the initial certainty factor is low.

The metaknowledge updating scheme presented here has all these features and is based on two functions that map the current certainty factor to a new one (Figure 6). The first function deals with confirmatory evidence that causes upward revision while the second one deals with contradictory evidence that causes downward revision. When a surrogate with a keyword is retrieved successfully, the revised certainty factor is given by $(1-r) * x + r * (f2(x))$, where f2 is the function dealing with upward revision, x is the original certainty factor, and r is the reliability of this type of information (typically 0.1). When the evidence supports a proposition to a degree of 0.7 (say), the mapping function is the weighted average of the two original functions in the ratio 0.7:0.3 (see Figure 7).

D. Heuristic Distance Measure and Learning Curves

A heuristic function is used to compute how much the current metaknowledge differs from the ideal metaknowledge for a particular description of the system state. The actual search sequence adopted by a user for a keyword-based search would depend on his metaknowledge; individual document bases would be scanned in decreasing order of certainty factors. These sequences are first computed from the current metaknowledge base for all search-pairs (user, keyword).

The ideal search sequences, on the other hand, are obtained from the current document base. For reasons explained earlier, documents in the simulated environment are augmented with the relevance factor assignments they would have elicited from users reading them. The best sequences for conducting keyword-based searches are obtained from this information.

The two sets of search sequences are now compared. If the two search sequences for a given search-pair are similar, the distance between them is small. One measure of disorder in a list is the number of exchanges between adjacent list elements required by a bubble-sort procedure to derive an ordered list. In this case the initial list is an actual search sequence derived from the metaknowledge, and the final ordered list is the ideal search sequence obtained from the document base. The measure of disorder of all the search-pairs are added together in order to obtain the total distance measure between the current and ideal metaknowledge patterns.

The heuristic distance is measured at the beginning of each simulation and after each measurement cycle of ten transactions. For the simulation results shown in Figures 8-11, 450 transactions (commands) were executed and a total of 46 measurements were made. Though the learning heuristics for these simulations were kept unchanged, different office models and usage patterns were employed. The graph of the distance measurements as a function of the number of transactions produces the learning curve for a particular office environment and particular set of learning heuristics. Properties of these graphs, such as time-constants and steady-state values, are being used to evaluate the performance of the heuristics in order to derive meta-heuristics.

V. THE MULTI-LAYERED LEARNING SYSTEM MODEL

Buchanan <u>et al</u>. [7] have proposed a general model for a learning system (LS) based on four functional components, the performance element, the learning element, the critic, and the instance selector. Each component has bidirectional communication links to a blackboard containing the knowledge base as well as all temporary information used by the learning system components.

The performance element is responsible for generating an output in response to each new stimulus. The instance selector draws suitable training instances from the environment and presents them to the performance element. The critic evaluates the output of the performance element in terms of a performance metric and suggests adjustments to the learning element, which makes appropriate changes to the knowledge base. The LS operates within the constraints of a world model which is the conceptual framework of the system with assumptions and methods for domain activity and domain-specific heuristics [8].

The world model can not be modified by the LS that uses it, but it may be altered by a higher-level system based on the observed performance of the LS. This system may itself be a learning system. Incremental refinement of the world model can thus be accomplished in a higher-level learning system (LS2) whose performance element is the learning system at the lower level (LS1). Several well-known LS's have been characterized using Buchanan's framework [7,9].

Dietterich _et al_. [10] have developed a simple model of learning systems that incorporates feedback from the performance element to the learning element. The included knowledge base is not specified as a blackboard. Their model has been used to examine the factors that affect the design of the learning element.

The MINDS two-level testbed for heuristic refinement has been mapped into an integrated framework, combining features of both of the above general models. This framework is shown below:

The User Layer

Goal: Learn to customize document searches for individual users in a dynamic setting.

The Upper Layer (LS2, Figure 13)

Purpose: Improve the performance of LS1 by selecting a good set of learning heuristics.

Environment: All command sequences, initial metaknowledge configurations and initial document distributions that comprise the training set.

Instance Selector: Chooses an interesting environment (a combination of command sequence, metaknowledge configuration, and document distribution) with help from the critic.

Critic: Evaluation. Uses meta-heuristics (presently supplied by a knowledge engineer) to analyze learning curves for LS1 with the current set of learning heuristics.
Diagnosis. Singles out heuristics that are not useful.
Therapy. Selects new heuristics to replace useless ones or suggests modifications for existing ones. Also suggests interesting environments for testing a new set of heuristics.

Learning Element: Redefines the current set of heuristics as recommended by the critic.

World Model: The LS1 world model along with the set of meta-heuristics for updating the LS1 heuristics, the method for evaluating the performance of LS1, and the scheme for heuristic updating.

The Lower Layer (LS1, Figure 12)

Purpose: For each user learn a good set of confidence factor assignments to predict the outcomes of all keyword-based searches of individually-owned document bases.

Environment: Set of all possible combinations of document distributions and user commands.

Performance Element: Uses the document metaknowledge (set of confidence factor assignments) to plan a document base search sequence for a user. Also executes some non-search commands which may change the document distribution pattern.

Instance Selector: The next command is read from a predefined sequence of commands. The document distribution pattern chosen to be part of the environment for the execution of a command is simply the document configuration arrived at after execution of the last command.

Critic: Evaluation. Examines the result of searching a target document base with the help of some learning heuristics.
Diagnosis. Determines that the document bases be searched in the order that would have proved most fruitful as determined from the results.
Therapy. Recommends increases and/or decreases of certainty factors.

Learning Element: Adjusts the certainty factors in the document metaknowledge base according to the critic's recommendation.

World Model: Representations of the document base, knowledge base and commands, the metaknowledge updating scheme, and the learning heuristics for the evaluation of results and recommendation of therapies.

VI. CONCLUSIONS AND PLANS

The learning testbed has provided a useful tool for developing heuristics for learning the document storage patterns from the perspective of individual users in a variety of distributed office environments. Offices are modeled on the basis of higher-level descriptions which are used to generate a distributed document base and a plausible transaction sequence for the office and to initialize the metaknowledge for each user.

Simulations are run for these office environments and the resulting learning curves are analyzed by a knowledge engineer, who tries to correlate the heuristics with the results. A better set of heuristics may be identified after several iterations. A search strategy is required for rapid convergence to an optimal set of heuristics for a given office environment. Part of the current research effort is to identify meta-heuristics that would help refine existing heuristics, as well as discover new ones.

REFERENCES

1. R. D. Bonnell, M. N. Huhns, L. M. Stephens, and U. Mukhopadhyay, "MINDS: Multiple Intelligent Node Document Servers," Proceedings IEEE First International Conference on Office Automation, December 1984, pp. 125-136.

2. _____, "A Distributed Expert System Approach to the Intelligent Filing System," USCMI Technical Report 84-17, University of South Carolina, November 1984.

3. J. B. Rothnie and N. Goodman, "A Survey of Research and Development in Distributed Database Management," Proceedings IEEE Third International Conference on Very Large Data Bases, 1977, pp. 30-44.

4. G. Hendrix and E. Sacerdoti, "Natural Language Processing: The Field in Perspective," BYTE Magazine, September 1981, pp. 304-352.

5. B. G. Buchanan and E. H. Shortliffe, "Reasoning Under Uncertainty," Rule-Based Expert Systems, Part Four, Addison-Wesley Publishing Co., Reading, MA, 1984.

6. D. B. Lenat, F. Hayes-Roth, and P. Klahr, "Cognitive Economy In a Fluid Task Environment," Proceedings of the International Machine Learning Workshop, R. S. Michalski,

(Ed.), Dept. of Computer Science, University of Illinois, Urbana, IL, 1983.

7. B. G. Buchanan, T. M. Mitchell, R. G. Smith and C. R. Johnson, Jr., "Models of Learning Systems," Encyclopedia of Computer Science and Technology, Vol. 11, J. Belzer, A. G. Holzman and A. Kent, (Eds.), Marcel Dekker Inc., New York, 1977, pp. 24-51.

8. P. H. Winston, "Learning Structural Descriptions from Examples," The Psychology of Computer Vision, McGraw-Hill, New York, 1975, pp. 157-210.

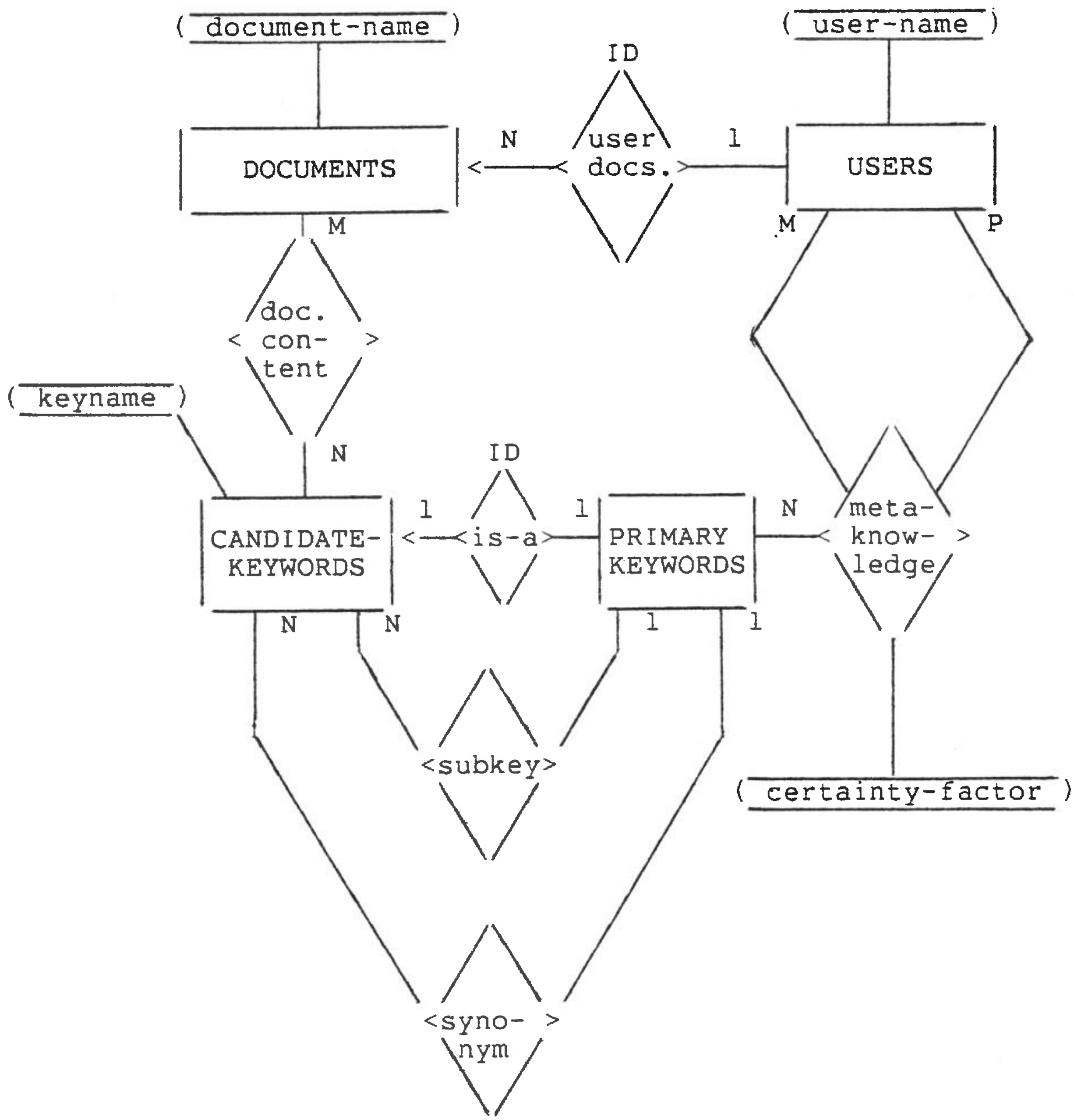
9. L. A. Rendell, "Conceptual Knowledge Acquisition in Search," Knowledge Based Learning Systems, L. Bolc (Ed.), Springer-Verlag, New York, to appear.

10. T. G. Dietterich, B. London, K. Clarkson, and G, Dromey, "Learning and Inductive Inference," Ch. XIV, The Handbook of Artificial Intelligence, Vol. 3, P. R. Cohen and E. A. Feigenbaum (Eds.), William Kaufmann, Inc., Los Altos, CA, 1982.

11. R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Co., Palo Alto, CA, 1983.

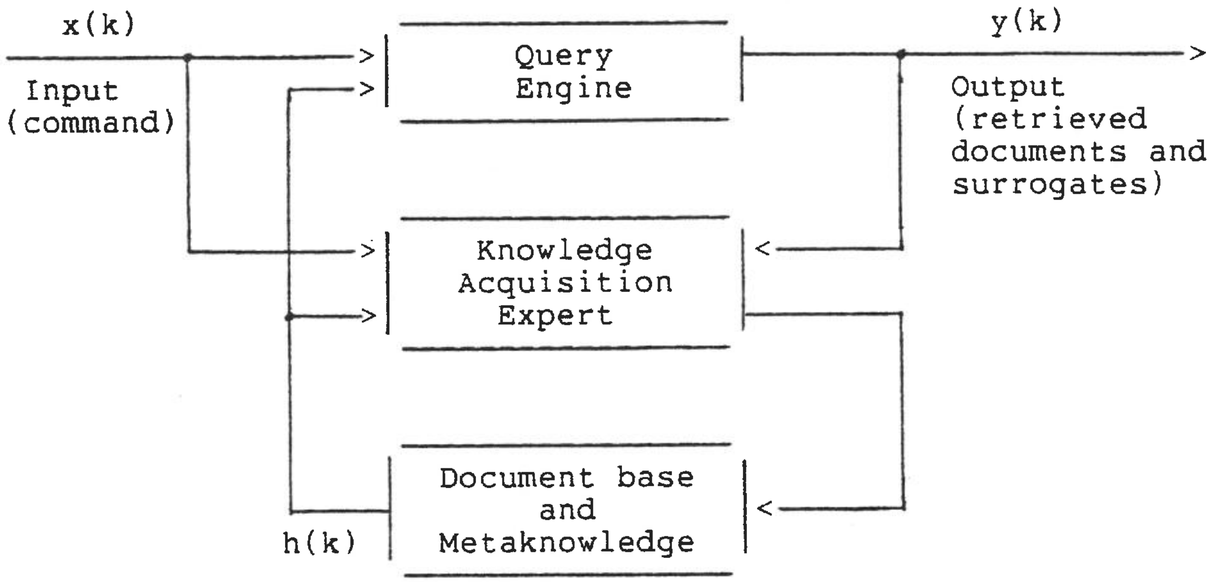
12. R. C. Schank and P. G. Childers, "The Learning Barrier," The Cognitive Computer, Ch. 8, Addison-Wesley Publishing Co., Reading, MA, 1984.

13. T. G. Dietterich and R. S. Michalski, "Learning and Generalization of Characteristic Descriptions: Evaluation Criteria and Comparative Review of Selected Methods," Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 1979, pp. 223-231.



Note: Most non-key attributes have been removed for clarity.

Figure 1. Entity-Relationship Diagram for the Metaknowledge of the System



Output function: $y(k) = q(\ x(k),\ h(k)\ )$

State function: $h(k+1) = a(\ x(k),\ h(k)\ )$

Figure 2. Block Diagram of System Dynamics

1. Set up heuristics
2. Set up data set
3. Start transaction
4. Update documents
5. Update metaknowledge
6. End transaction
7. If no. of transactions since last measurement is less than limit go to 3, else continue
8. Make measurement
9. If no. of measurements for this data set is less than limit go to 3, else continue
10. Evaluate measurements
11. If no. of data sets for this simulation is less than limit go to 2, else continue
12. Evaluate and compare observations
13. If performance of heuristics is not satisfactory, then go to 1, else STOP.

Figure 3. The Heuristics Refinement Cycle

```
((user1
  (doc27 (key13 (obj_user1 0.0)(obj_user2 0.1)(obj_user3 0.3))
         (key7  (obj_user1 0.7)(obj_user2 0.2)(obj_user3 0.5))
         (key5  (obj_user1 0.7)(obj_user2 1.0)(obj_user3 0.4)))
  (doc28 (key0  (obj_user1 0.1)(obj_user2 0.4)(obj_user3 0.3))
         (key11 (obj_user1 0.2)(obj_user2 0.1)(obj_user3 0.0))))
 .
 .
 .
 (user3
  (doc37 (key10 (obj_user1 0.6)(obj_user2 0.5)(obj_user3 0.7))
         (key14 (obj_user1 0.6)(obj_user2 0.3)(obj_user3 0.4)))))
```

Figure 4. Document Base Representation in the Testbed

```
((user1 (key0 (obj_user1 0.4) (obj_user2 0.4) (obj_user3 0.4))
        (key1 (obj_user1 0.9) (obj_user2 0.7) (obj_user3 0.7))
        .
        .
        .
        (key9 (obj_user1 0.2) (obj_user2 0.2) (obj_user3 0.5)))
 .
 .
 .
 (user3 (key0 (obj_user1 0.1) (obj_user2 0.6) (obj_user3 0.4))
        .
        .
        .
        (key9 (obj_user1 0.9) (obj_user2 0.0) (obj_user3 0.4))))
```

Figure 5. Metaknowledge Representation in the Testbed

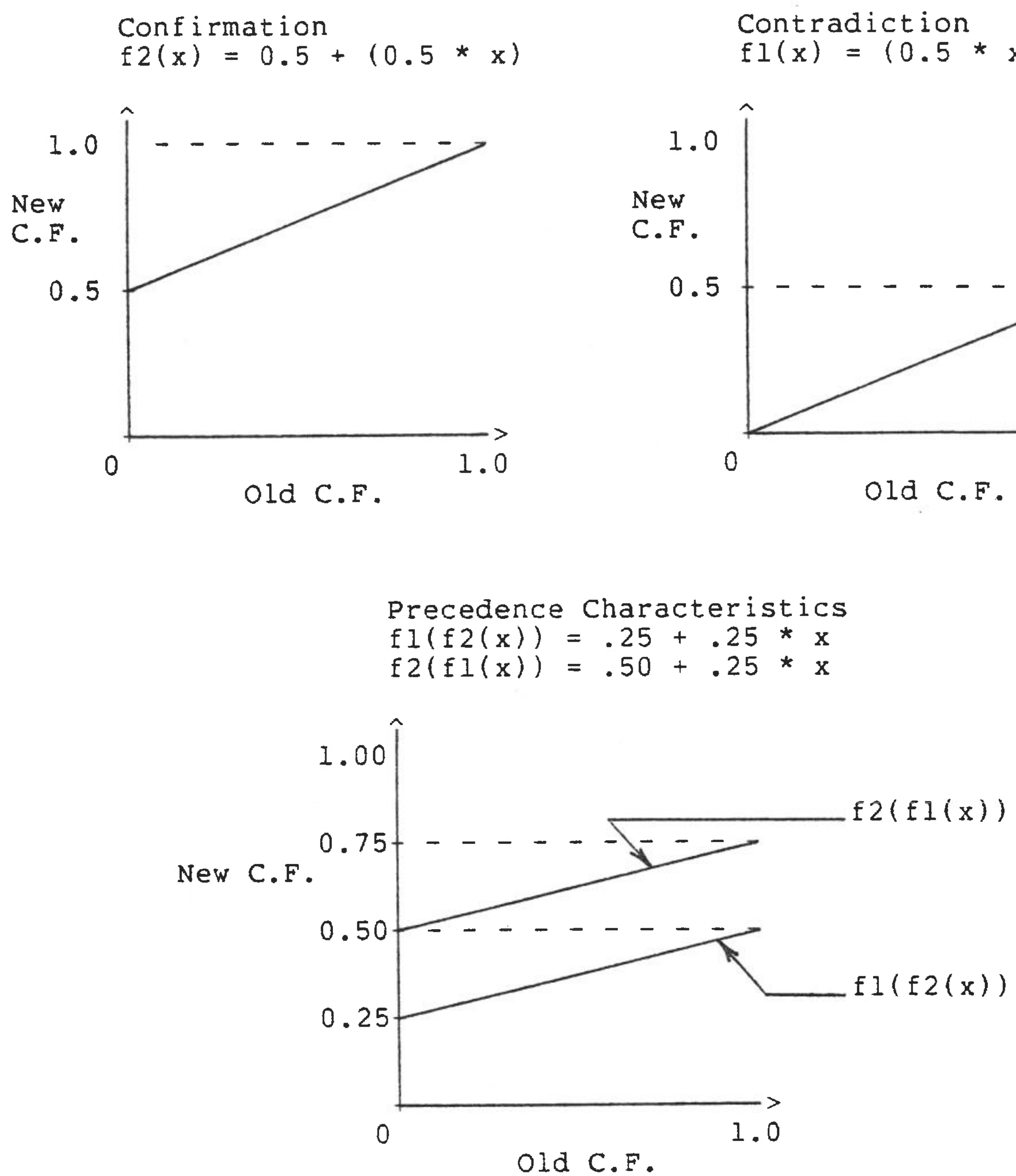


Figure 6. Update Functions for Metaknowledge Certainty Factor (C.F.) and Temporal Precedence Characteristics

Mapping rule for relevance, r = 0.7

$$f(x) = (r * f2(x)) + ((1-r) * f1(x))$$

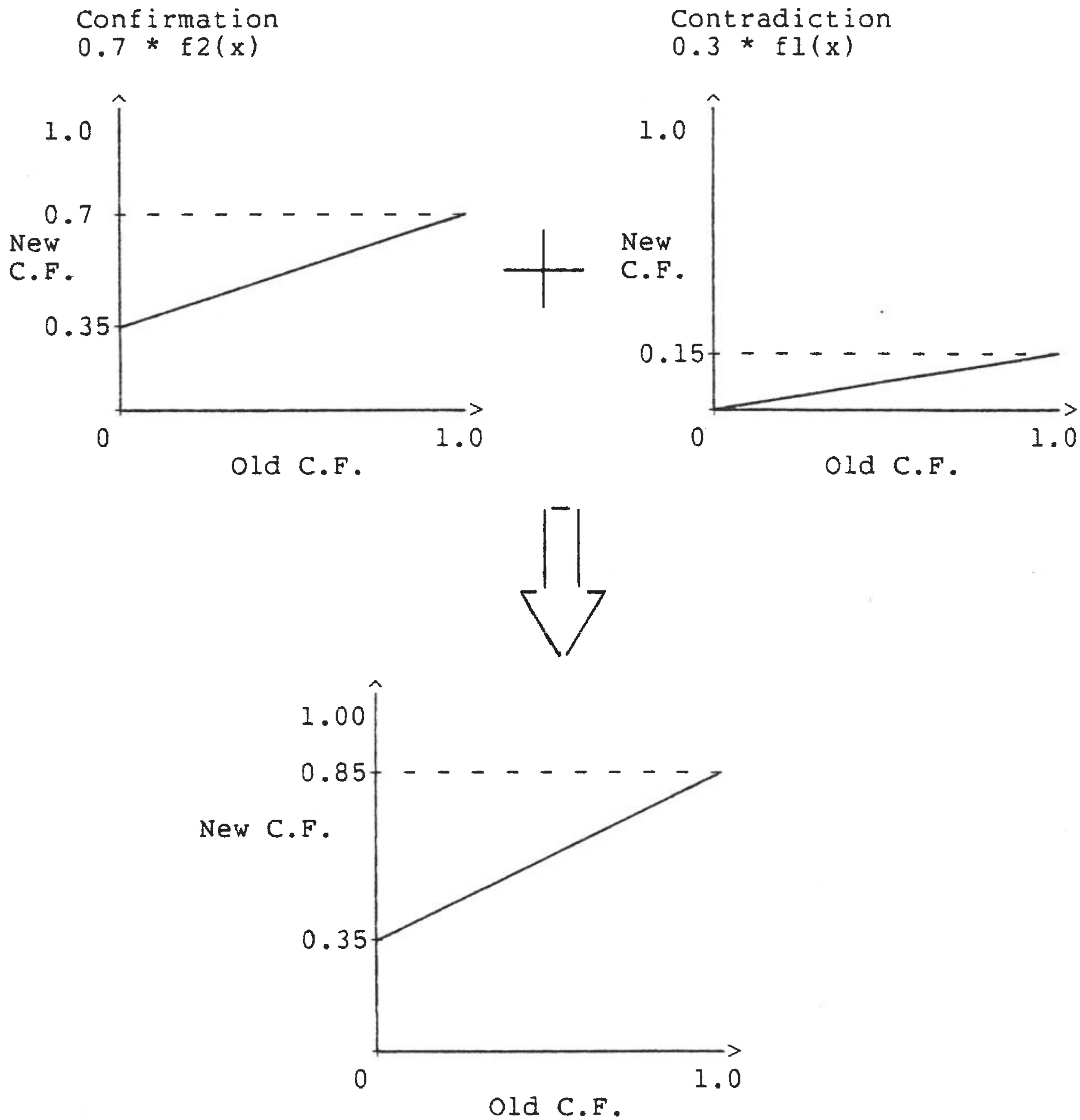


Figure 7. Updating Scheme for Metaknowledge Certainty Factors (C.F.)

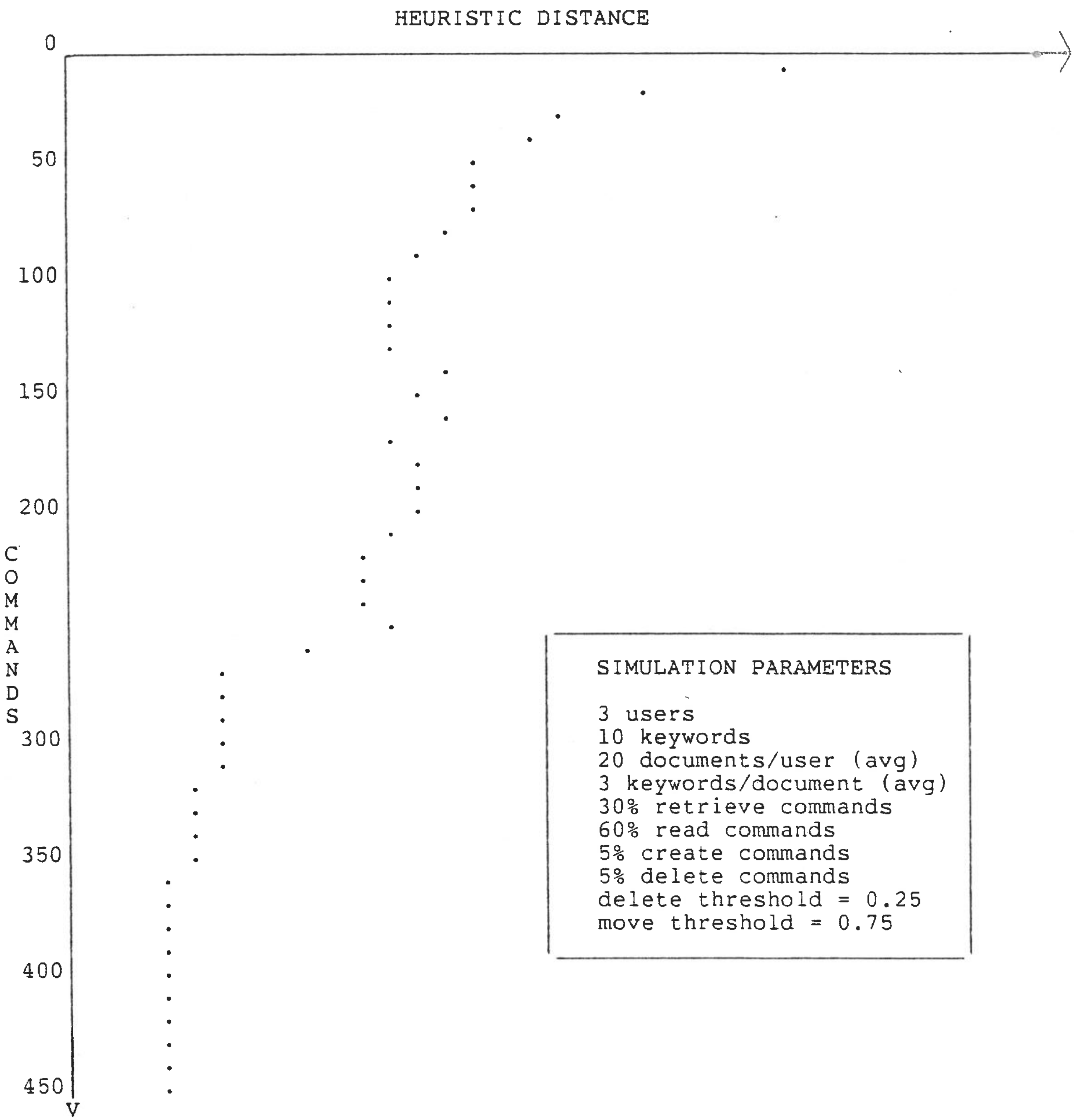


Figure 8. Learning (Decrease in Heuristic Distance) As Metaknowledge Is Accumulated While Processing Document Commands

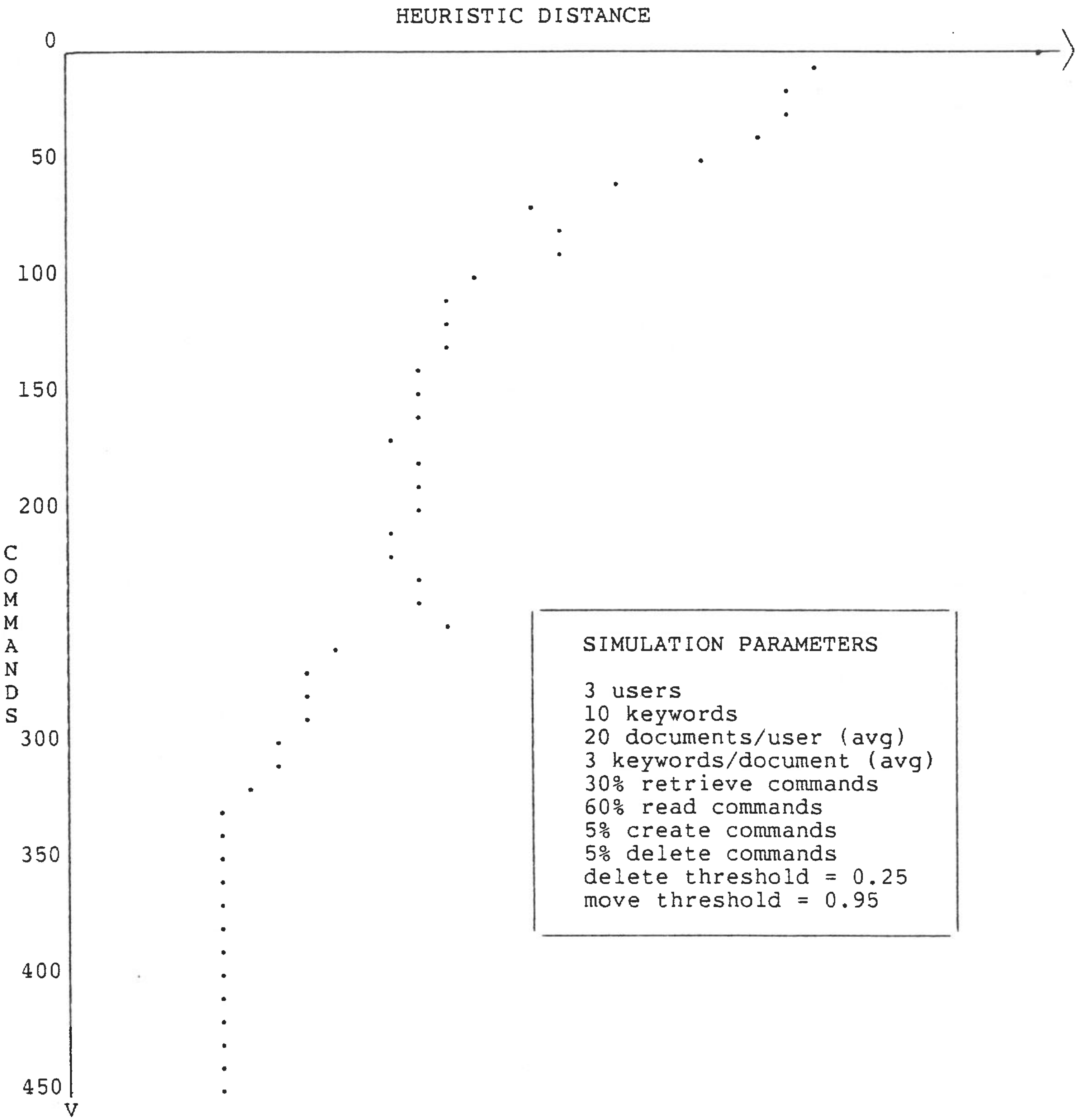


Figure 9. Learning (Decrease in Heuristic Distance)
As Metaknowledge Is Accumulated While
Processing Document Commands

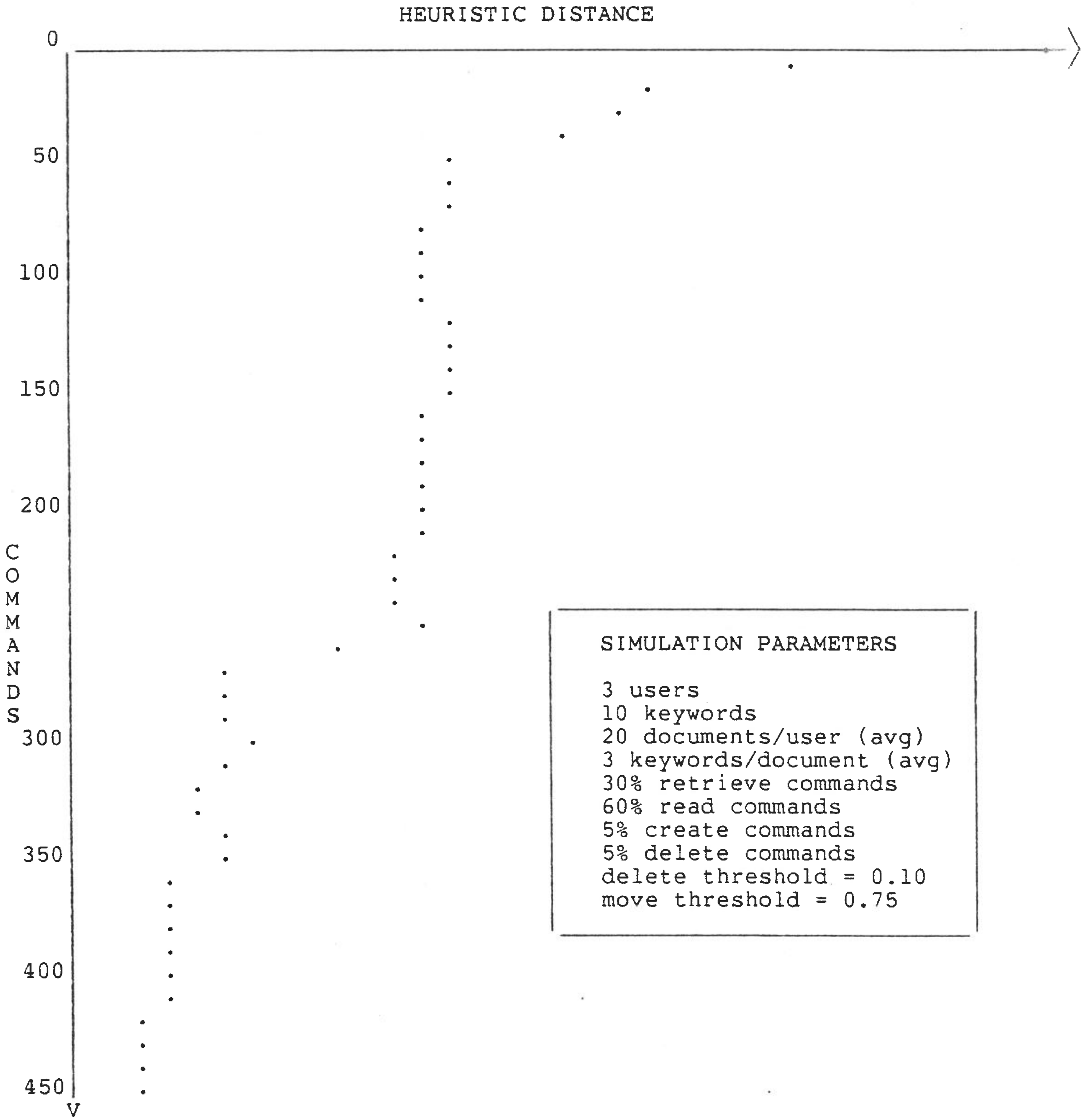


Figure 10. Learning (Decrease in Heuristic Distance) As Metaknowledge Is Accumulated While Processing Document Commands

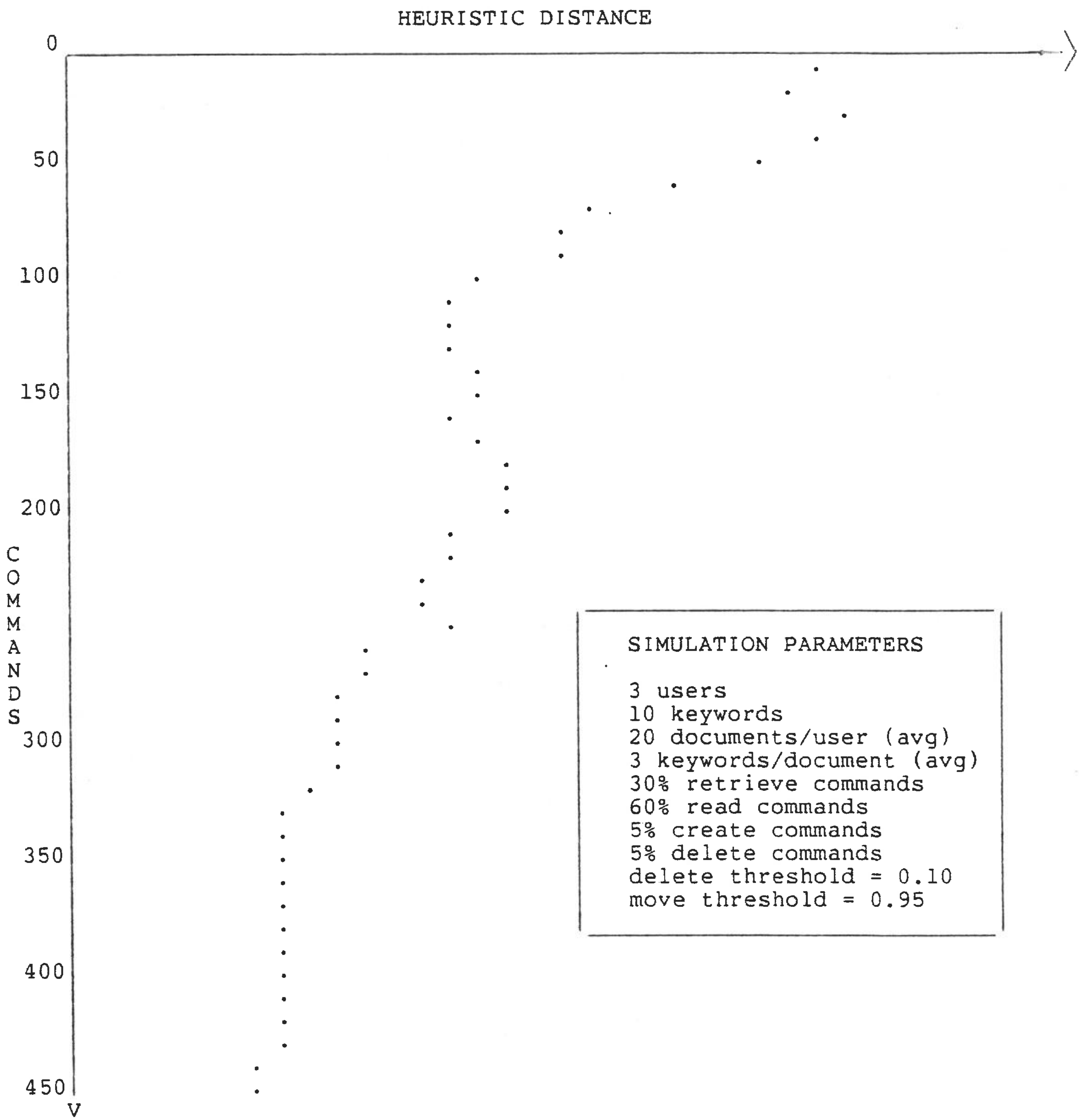
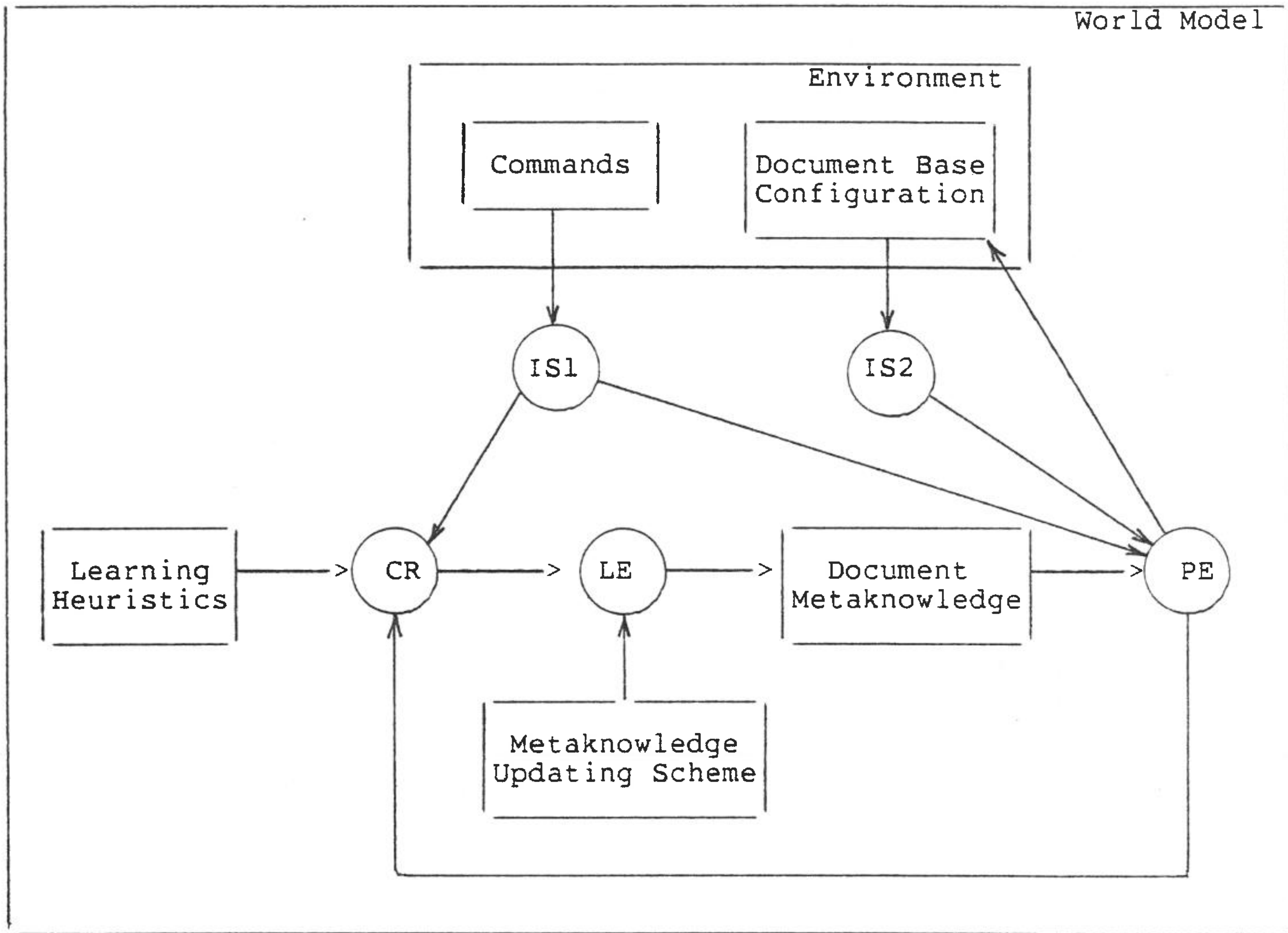
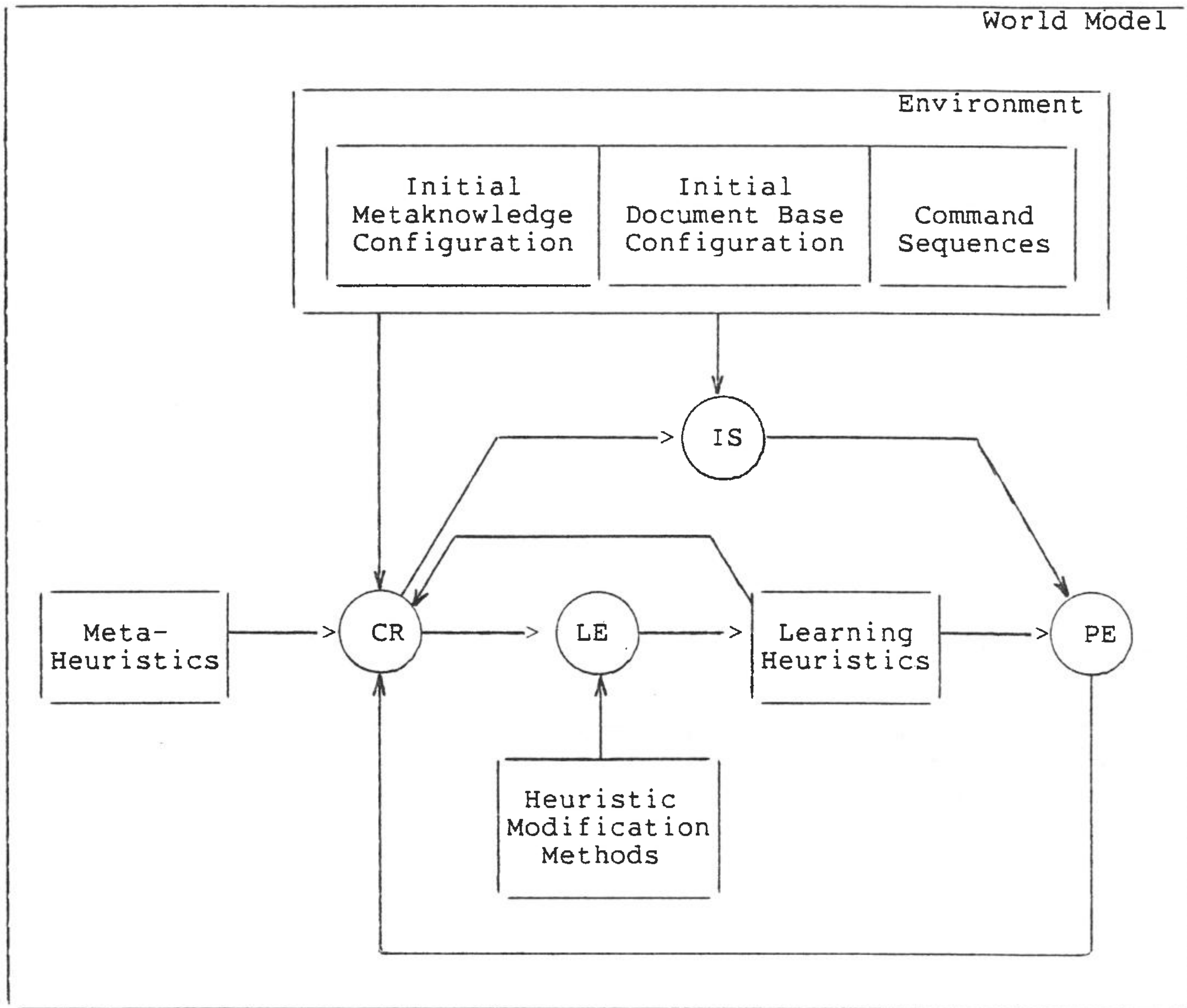


Figure 11. Learning (Decrease in Heuristic Distance) As Metaknowledge Is Accumulated While Processing Document Commands



CR - Critic
LE - Learning Element
PE - Performance Element
IS1 - Instance Selector for commands
IS2 - Instance Selector for document base configuration

Figure 12. Learning System LS1 (Lower Layer)



CR - Critic
LE - Learning Element
PE - Performance Element
IS - Instance Selector

Figure 13. Learning System LS2 (Upper Layer)