

# Learning Plan Libraries for Case-based Plan Recognition

Boris Kerkez

Department of Computer Science and Engineering  
Wright State University, Dayton, OH  
bkerkez@cs.wright.edu

## Abstract

This paper addresses the indexing and retrieval issues in the context of the case-based plan recognition. The indexing and storage mechanisms utilize the knowledge about planning situations that enable the recognizer to focus its search to a subset of the plan library containing relevant past plans. A two-level abstract indexing scheme, along with the incremental construction of the plan libraries, may significantly reduce the retrieval efforts of the recognizer. Adding a third level of indexing may also improve the retrieval, but it may be computationally too expensive for some planning domains. Experimental results show the next action prediction accuracy with and without utilization of the two-level indexing scheme.

## Introduction

The field of plan recognition is concerned with inferring the intentions of some planning agent. Some of the earliest plan recognition systems were focused on the story understanding (Wilensky 1980), question answering (Pollack 1986), and natural language dialogue understanding (Allen and Perrault 1980), while Kautz laid out theoretical foundations of the field at a later time (Kautz 1991). Traditionally, observed plans are considered to be recognized when they can be matched to some existing plan from the plan library. The recognizer's plan library stores the plans that the planning agent may pursue. Real-time recognition systems (recognition is performed during the plan execution) are also concerned with the planner's intentions that are more localized. Predicting the next planning action is an example of such localized prediction of the planner's intent.

During *intended* plan recognition, the planning agent is aware of the recognizer and tries to assist (cooperative) or obstruct (adversarial) the recognition task. An example of intended recognition is natural language dialogue, where the speaker intends to convey the plan to the listener via utterances. During *keyhole* plan recognition, the recognizer has no input from the planning agent and recognition is

performed from observations of the planning agent alone. All information about the agent's plan execution is obtained through keyhole-like observations of the planner's behavior.

Many plan recognition systems recognize observed plans by simply matching the observed partial plans with the plans contained in the plan library. Most plan recognition systems operate with a complete plan library that contains all of the possible plans the planning agent may pursue. There are several problems with complete plan libraries. First, plan libraries are often constructed in advance, and often by a human knowledge engineer. Second, complete plan libraries may contain extraneous plans (i.e., plans that are never pursued), which can impact the efficiency of the recognizer (Lesh and Etzioni 1996). Finally, enumeration of all possible plans may not be possible in some domains, either because of the domain complexity or because of a lack of the domain knowledge.

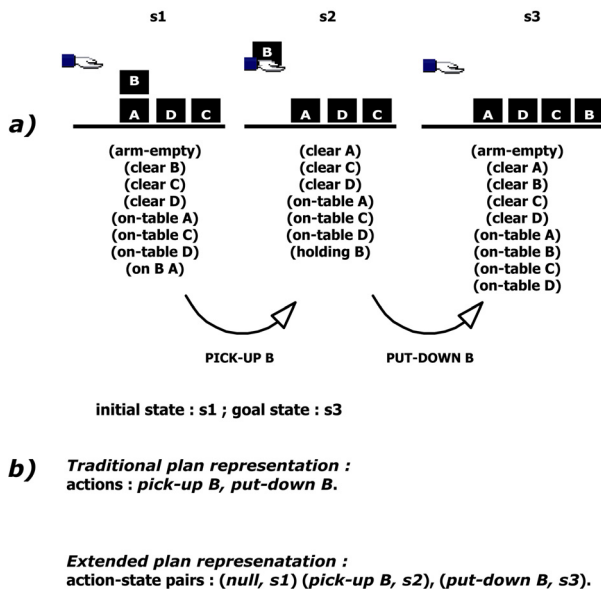
The research presented in this paper is concerned with *keyhole* plan recognition with *incomplete* plan libraries. This incompleteness introduces additional issues, because plans that are observed by the recognizer may not be present in the plan library. Newly observed plans must be incorporated appropriately into the plan library, so that they are available for recognition at subsequently observed planning steps. The recognizer described in this work incorporates a case-based approach (Riesbeck and Schank 1989, Kolodner 1993) into plan recognition in order to learn novel plans. Although Bares and his colleagues also deal with a case-based plan recognizer (Bares *et. al.* 1994), their approach to indexing and retrieval is quite different than that presented here.

Unlike most traditional recognizers that reason in terms of actions performed by the planner, the case-based recognizer reasons in terms of situations in which the planner finds itself during the plan execution. An effective two-level indexing scheme is used to provide efficient access to previously observed situations. This indexing scheme enables the recognizer to form predictions even in light of novel planning actions and to significantly increase its local prediction accuracy as shown by experimental results.

The next section describes representational and indexing schemes in the presented plan recognition system. Abstract indexing techniques are described in more detail in Section 3, while Section 4 introduces an extension of the indexing scheme and discusses its benefits as well as its drawbacks. Experimental results are presented in Section 5. The last section concludes the paper by summarizing the accomplishments.

### Situational Indices as Reminding Elements

When dealing with a plan recognition system with an incomplete plan library, the recognizer needs to learn the newly observed plans and to store these plans in a plan library. Newly observed plans need to be indexed appropriately, so that they are retrieved by the recognizer

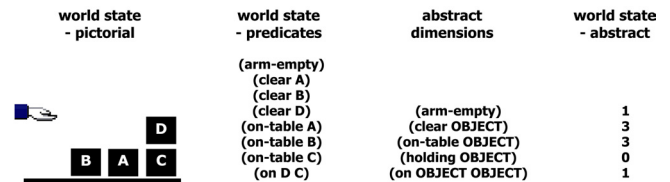


**Fig. 1.** a) An example of a simple planning episode from the blocksworld planning domain. b) Two different views of an observed plan.

when similar plans are observed in the future. Therefore, storage and retrieval processes are mutually dependent and must be considered together. This dependency is a defining characteristic of case-based systems that create indices at storage time. The plan recognition system described in this work integrates case-based indexing, retrieval and adaptation processes to form predictions in light of novel planning actions. The retrieval of known planning episodes is facilitated by the case-based reminding processes.

In order to study the properties of the plan recognition, we modified PRODIGY (Carbonell *et al.* 1992) state-space planner's execution cycle to display not only actions executed during the plan, but also the states of the world in which the planner finds itself after each action application.

The world states represent planning situations that are used as indices for the reminding processes of the case-based plan recognizer. At each recognition step during the plan execution, the recognizer observes the action performed by the planner, as well as the state of the world reached by the performed action. Figure 1 illustrates the plan representation extended with intermediate state knowledge in a simple blocksworld planning domain. Instead of reasoning only in terms of actions performed by the planner, the case-based plan recognizer uses the state caused by the action and attempts to find similar situations that were encountered in the past. If such situations are found in the case library, previous intentions of the planner in similar situations may be used to explain the current intent of the planner. This effectively enables the recognizer to form predictions even when new actions are observed, because new actions may lead to situations that were observed in the past.



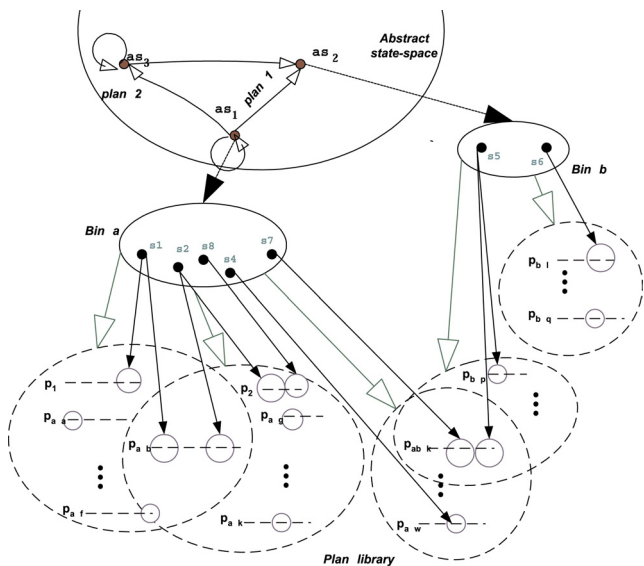
**Fig. 2.** An example of representational scheme from the blocksworld planning domain.

Therefore, utilizing the knowledge about both planning actions and planning situations can be very beneficial for a recognizer with an incomplete plan library. On the other hand, knowledge about planning situations increases the complexity of the recognition task. The number of different planning situations is large even for the smallest planning domains because of the inherent complexities of the planner's state-space (Kerkez and Cox 2001). In order to cope with the complexity of the state-space, the case-based plan recognizer employs an indexing scheme that utilizes the abstraction of world states into non-negative integer vectors. Figure 2 shows an example state from the blocksworld domain and its abstract representation. Each abstract dimension represents the number of occurrences of state predicates of a given type within the state. Because the state in Figure 2 has three instances of the *clear* state predicate, the second value in its abstract dimension (labeled *clear*) is also three.

### Indexing and Retrieval Using State Indices

The abstraction of world states into non-negative integer vectors provides the first level of a two-level indexing scheme for the retrieval of past situations. Concrete world states are indexed by their abstract representation into

structures called *bins*. All states within a single bin have the identical abstract representation. Note that the mapping from the abstract states to concrete states is one-to-many, as one bin may contain one or more concrete world states. Also, all states with identical abstract representations are guaranteed to be in the same bin. Once a correct bin is located, concrete world states from that bin are selected as the second level indices. Concrete states, in turn, point to plans in the plan library in which they are contained. Figure 3 illustrates the abstract indexing scheme graphically.



**Fig. 3.** Indexing and storage structures. Abstract states ( $as_i$ ) point to bins, containing world states ( $s_j$ ). World states in turn point to past plans ( $p_k$ , dashed straight lines) in which they are contained. Solid lines with arrows represent indexing links.

Notice that the abstraction mechanism can also be applied to the planning actions. In this case, an abstracted action is created by replacing the actual arguments of actions with their leaf types from a given type hierarchy. For example, the concrete action “*stack blockA blockB*” from the blocksworld domain is abstracted into “*stack OBJECT OBJECT*”, because *OBJECT* is a leaf type of all blocks in this planning domain.

The exact complexity associated with this indexing scheme depends on the characteristics of the domain state-space. The distribution of concrete states in their corresponding bins depends on the number of both abstract and concrete world states. Given a large number of concrete world states, the number of abstract states would have to be reasonably large in order not to overfit the bins with too many states. Experimental results from the logistics planning domain with three cities indicate that the distribution of concrete states in the bins is feasible for the recognizer to handle. After about 30,000 planning steps, the recognizer observes around 400 abstract states and more than 15,000 concrete states. The maximum bin

contains about 200 concrete states, the minimum bin contains a single state, while average bin size is only about 40. The recognizer can therefore focus its attention to a single bin and consider only a small subset of all possible states in the bin

In certain planning domains it may be possible to extend the indexing scheme to further differentiate the world states within a single bin. This approach is discussed in detail in the next section.

## Equivalence-Class Indexing

Although the distribution of concrete world states into bins is very helpful in reducing the search during retrieval in the blocksworld and the logistics domains (Kerkez 2001), the question still remains whether this indexing approach is suitable for state-space planners in general. It is certainly possible for a bin to index a large number of different concrete world states. In cases where the average bin size is large, the retrieval efficiency may be impacted by having to consider too many indices. To cope with the problem of large indexed bins, a representation change technique (Amarel 1968, Fink 1999) can be applied to the states of the planner’s world. This representation is graph-based, where the states are represented as directed graphs with loops. These graphs are called *directed nets*. Let

$$p_S^i = \{p \mid p \text{ is a predicate} \wedge S \text{ is a state} \wedge p \in S \wedge |\text{args}(p)| = i\}$$

a set of predicates in world state  $S$ , which have  $i$  arguments. Given a world state

$$S_i = \{p_{S_i}^0, p_{S_i}^1, \dots, p_{S_i}^q\}$$

where

$$\text{args}(p_{S_i}^j) = \{ \bigcup_{k=1}^{(p_{S_i}^j)^* j} o_k \mid o_k \text{ is a domain instance} \}$$

a set of all domain instances that are arguments of predicates with arity  $j$  in state  $S$ , the vertices of a graph representing a world state  $S$  can be defined as

$$V(G_S) = \{ \bigcup_{l=1}^q \text{args}(p_{S_i}^l) \} \cup p_S^0$$

An edge from one vertex to another one indicates that the former appear before the latter in the list of arguments of the same predicates. Formally,

$$e_{o_i, o_j} \Leftrightarrow \exists p \mid p \text{ is a predicate} \wedge \{o_i, o_j\} \in \text{args}(p)$$

Figure 4 shows a pictorial example of a representational change from a predicate representation consisting of ground literals into the state graph. Notice that edges can be loops when a state predicate has only one argument. The notion of edges can be extended to the cases where

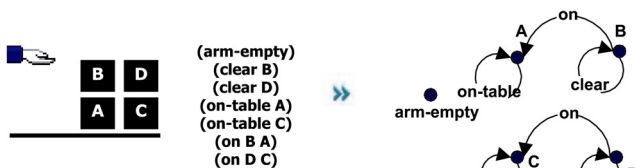


Fig. 4. An example state graph representation from the blocksworld planning domain.

state predicates have more than two arguments. By applying a hypergraph representation an edge can be adjacent to more than two vertices. However, the current implementation of the recognition system is limited to the state predicates with at most two arguments. Because any given hypergraph can be represented as an equivalent bipartite graph, this technique is applicable to domains with three or more arguments.

The change of representation comes into play when a single indexing bin contains *structurally* different states. To illustrate this, consider Figure 5, which describes two structurally different world states having the same abstract representation and thus belonging to the same bin. State abstraction alone is not able to capture the differences among structurally different world states in the same bin. However, once the states in the same bin are transformed into their corresponding state graphs, the structural

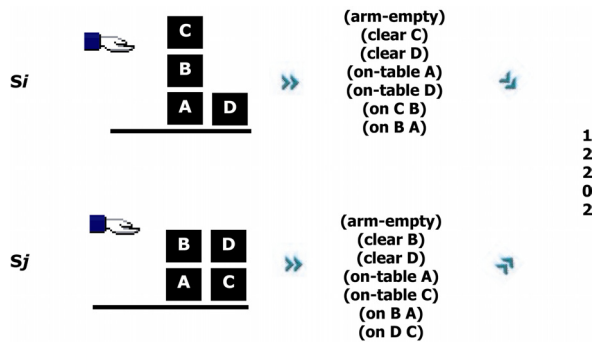


Fig. 5. An example of two structurally different world states  $s_i$  and  $s_j$  with identical abstract representations.

differences are evident. Essentially, two states will be structurally identical if and only if their state graphs are isomorphic (Kerkez 2001). The graph isomorphism is a one-to-one and onto mapping that maps the vertices of one graph onto the vertices of another graph while preserving the edges. Figure 6 shows the two states from Figure 5 in the state-graph representation; it can easily be seen that the two graphs are not isomorphic.

Transformation of a world state in the predicate form into its state graph is obviously linear in the number of vertices and edges of the state graph. Given such little overhead, the benefits introduced by this representational change have a potential to focus the retrieval of past

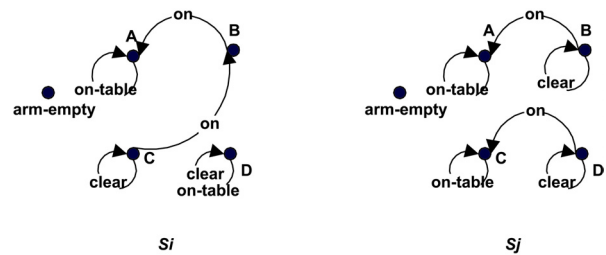


Fig. 6. State graphs for states  $s_i$  and  $s_j$  from Figure 5 with identical abstract representations.

situations to a smaller subset of relevant world states. This is because world states within the same bin can be further separated into groups of structurally equivalent states. Graph isomorphism is an equivalence relation that naturally partitions the bins into equivalence classes containing only structurally equivalent states. By comparing the two state graphs at storage time, the recognizer can determine whether they are isomorphic and thus structurally equivalent. This comparison facilitates efficient search for a matching equivalence class within a bin, because state graphs of all states in a single equivalence class are isomorphic. The recognizer will be

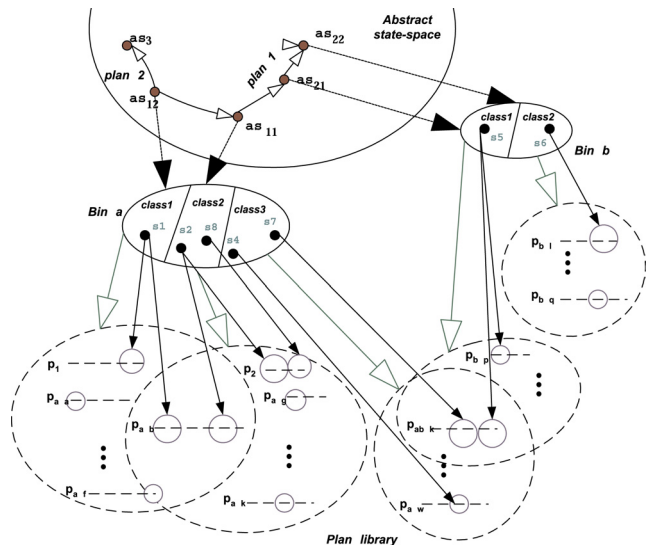


Fig. 7. Indexing and storage structures with equivalence classes induced by graph isomorphism relation

able to focus its retrieval efforts on past plans pointed to by a single equivalence class containing only structurally identical past situations. Figure 7 shows the indexing and storage structures extended to the equivalence class concept. Note that bins are further segregated compared to Figure 3. Also, abstract states from Figure 3 now index different equivalence classes instead of bins.

The main problem with equivalence class groupings is that the complexity of the graph isomorphism problem may be too large for the recognizer to handle. This is true

whether the two compared state graphs are isomorphic or not. The computational complexity of the graph isomorphism problem is unfortunately not known. It is not proven to be an *NP*-complete problem, and it is also not known to be in the class *P*. This problem is often speculated to be in the intermediate complexity class *NPI*, particularly because both the counting and decision versions of this problem are equally hard, a property not shared by *NP*-complete problems. However, polynomial-time graph isomorphism algorithms exist for several classes of graphs. Examples are graphs with bounded degree (Luks 1982), trees (Aho *et al.* 1974), and partial *k*-trees (Bodlaender 1990), to mention a few. Most of these polynomial isomorphism algorithms are, however, not applicable in practice because of large constant multipliers. The exceptions are planar graphs and circular-arc graphs (Colbourn and Booth 1981), for which efficient isomorphism algorithms exist.

The graph isomorphism problem can also help us understand the complexities associated with the substitution of arguments for state predicates in the predicate representations consisting of the ground literals (Velo 1994). The isomorphism mapping of the vertices of one graph onto another graph gives us the desired argument substitutions. Given two vertices  $v1$  and  $v2$  belonging to state graphs  $G1$  and  $G2$  respectively, let  $f$  be the isomorphism mapping between the two graphs. If  $f(v1)=v2$ , then the domain object corresponding to the vertex  $v1$  from the first world state substitutes for the domain object corresponding to the vertex  $v2$  from the second world state under the substitutions induced by the isomorphism mapping.

The analysis of substitution complexity is facilitated by the analysis of abstract state vectors. Because the edges of the state graphs are labeled, the isomorphism mapping is equivalent to a matching of the subgraphs of the state graph induced by the groups of edges with same labels. The maximum number of possible matches in an edge group is determined by the maximum value of the abstract vector dimension with the same label as the edge labels within the group. In general, given a state vector

$$v_s = \begin{bmatrix} d_1^{v_s} \\ \dots \\ d_k^{v_s} \end{bmatrix}, \text{ with}$$

$$v_{\max} = \begin{bmatrix} m_{d_1}^{v_{\max}} \\ \dots \\ m_{d_k}^{v_{\max}} \end{bmatrix}, \text{ where } m_{d_i} = \max\{d_i^v, \forall v, \forall i \in \{1, \dots, k\}\}$$

i.e.,  $m_d$  represents the maximum value of an abstract dimension  $d$  over all abstract states. For example, in the blocksworld domain with only four blocks (Figures 4 and

5), the maximum possible value of the *on* dimension is three, because no more than three blocks can be stacked on top of each other. The maximum number of possible argument substitutions  $\sigma$  when finding the isomorphism mapping is

$$|\sigma| = \prod_{i=1}^k (m_{d_i})!$$

In cases where all possible argument substitutions must be found and without biases induced by domain knowledge, the overhead introduced by finding all isomorphisms of two state graphs is computationally too expensive. Consider for example two different world states from the blocksworld planning domain, both of which include 50 different blocks that are *on-table* and *clear*. Any of these blocks from one state may be substituted for any of 50 blocks from another state, and the total number of possible substitutions among only this one group of blocks will be greater than  $(50!)$ . The total overhead introduced by argument substitutions depends on maximal abstract state  $v_{\max}$  and is thus domain dependent.

It is possible to utilize knowledge about the domain theory to assure that the argument substitution scheme for state predicates is feasible. When all state graphs in a given planning domain belong to a certain class of graphs for which efficient isomorphism algorithms exist, then equivalence classes can be efficiently used to group similar past situations inside the bins. If we consider the logistics planning domain without the static state predicates<sup>1</sup>, then the state graphs are always planar. Finding isomorphisms for planar graphs is efficient, and it may help focus the retrieval of past situations in the logistics domain. However, this approach does not generalize to other planning domains, because the state graphs of these domains may not all be planar.

Another approach to cope with the need for the argument substitution is to limit the substitutions to a subset of all domain objects. A natural choice is to limit the substitution to include only the arguments of a planning action for agent's next action prediction. Because the number of arguments of a typical planning action is small, substitutions for the action arguments can be found much easier. In terms of the graph state representation, we attempt to find the isomorphism mapping of two graphs induced by the vertices that are the arguments of the considered action. Because the induced graphs are much smaller than the complete state graphs, finding isomorphism among induced graphs will be simpler. Initial experimental results (Kerkez and Cox, unpublished) indicate that argument substitutions found with the help of induced graphs along with domain independent heuristics

<sup>1</sup> *Static* state predicates remain unchanged during the execution of a set of problems; *dynamic* state predicates may change during the execution.

can significantly improve the prediction accuracy of case-based plan recognizer.

## Experimental Results

To determine the benefits of the abstract-level indexing scheme, we performed plan recognition experiments in the logistics domain. In this domain, cities are located on islands and are accessible only by planes. Each city has a post office and an airport, as well as one or more trucks. The main objective is to construct plans that will transport the domain objects to desired locations, which may be post offices and airports in different cities. Figure 8 shows a simple example of the logistics scenario with two cities. Static state predicates remain unchanged during the execution of a set of problems. Figure 9 shows the ground literals representing the state in Figure 8.

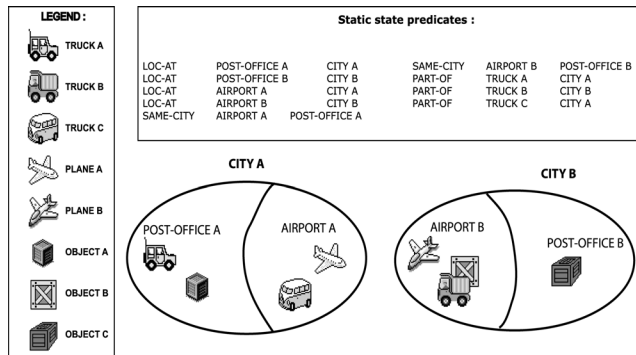


Fig. 8. A simple example of the logistics planning domain with two cities.

The experimental evaluation focused on randomly generated planning problems with 3, 5, and 7 cities in the logistics domain. Evaluations presented in this paper show results for problems with three cities. This is because asymptotic behavior of the recognizer can clearly be observed, given a smaller state-space in the case of three cities. Next, two different problem sets with 5000 problems each were randomly generated for the 3 cities logistics domain. The random generator for the problems in each of these two problem sets was initiated with a

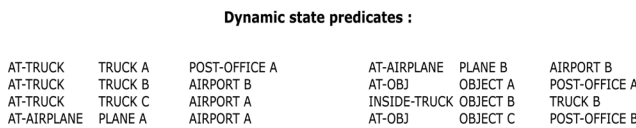


Fig. 9. Dynamic state predicates for the state in Figure 8.

different random seed. Generated problems were executed in the PRODIGY state-space planner that generated solutions to about 80% of the input problems. The reason for the 20% planning failure rate is due to the fact that

some random problems are impossible to be solved. In these situations the planner returns from its execution cycle without creating a plan once appropriate maximum execution time threshold is reached.

The solved planning problems and the generated solutions were then given as the input into the case-based plan recognition system. The system simulates real-time plan execution by parsing the input data and by sending the planning steps to the recognizer one at the time. The system is concerned with predictions of the local planning behavior. In particular, predictions of actions that are likely to be pursued next by the planning agent. Predictions are made at random by randomly choosing the next action from a set of all possible next planning actions. To illustrate the benefits of the abstract-level indexing scheme employed by the case-based plan recognizer, we present the averaged recognition results of the two problem sets in three city logistics problem sets. As a baseline test, we used a linear indexing scheme that randomly chooses a planning action from all known (i.e., previously observed) actions without the abstract indexing scheme. The baseline performance is compared to the recognizer that utilizes abstract-level indexing described in Section 3. Although the recognizer still forms predictions at random, this time choices for the next actions are limited by the actions performed in past situations found in a single indexing bin. Figure 10 shows the number of correctly predicted actions at both abstract and concrete levels after about 3,800 different cases and more than

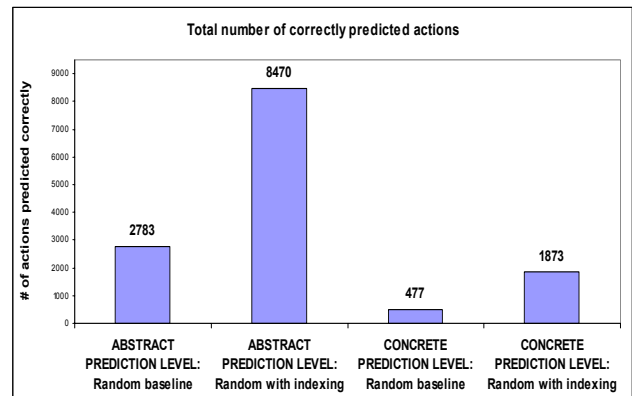


Fig. 10. Total number of correctly predicted actions at both abstract and concrete levels; with and without abstract indexing.

30,000 recognition steps. Clearly, the use of abstract indexing scheme significantly increases the next action prediction accuracy of the case-based plan recognizer. The percentage of correct predictions over time for abstract and concrete action predictions is shown in Figures 11 and 12 respectively. Prediction percentages are initially chaotic due to the sparse plan libraries. After a few thousand observed planning steps, the prediction accuracy percentages reach their asymptotic value in both cases. The

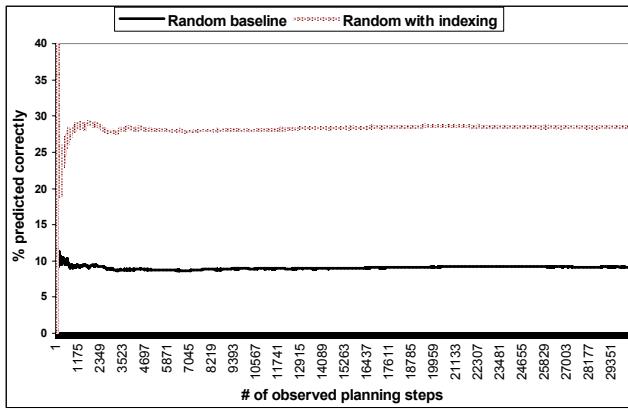


Fig. 11. Percentages of correctly predicted *abstract* actions, with and without abstract state indexing.

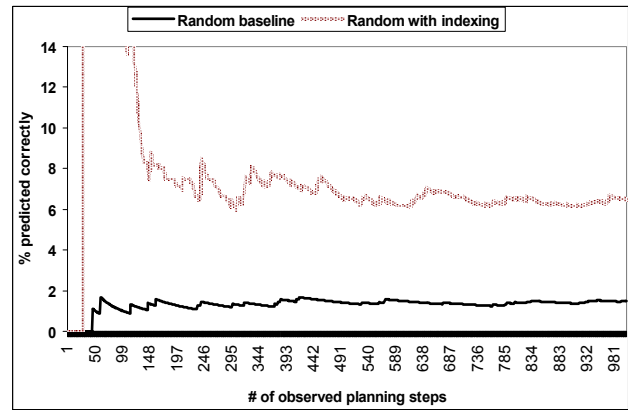


Fig. 13. Percentages of correctly predicted *concrete* actions for the first 1000 planning steps observations, with and without abstract indexing.

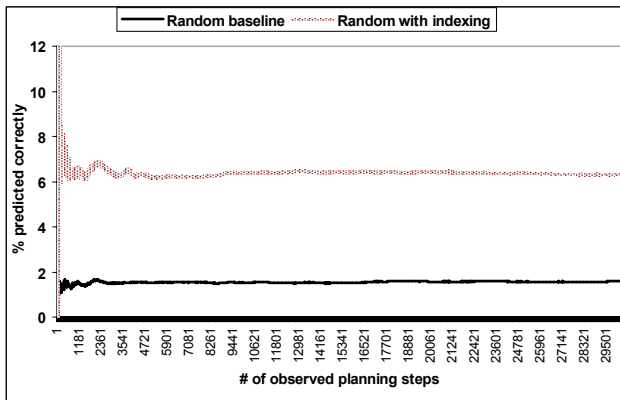


Fig. 12. Percentages of correctly predicted *concrete* actions, with and without abstract state indexing.

initial non-asymptotic prediction percentage accuracy for the concrete actions is shown in Figure 13. The results plotted are for the initial 1,000 planning steps.

Therefore, the abstract-level indexing scheme focuses the prediction choices and significantly improves the prediction accuracy over an extended recognition period. While the prediction accuracy of abstract-level indexing scheme is much greater than the baseline accuracy, there is much room for the improvement, because only about 6% of action predictions at the concrete level result in successful predictions. Replacing the random choices for action prediction with intelligent ones and utilizing the argument substitutions significantly improves the percentages of correct predictions on both levels of abstraction.

## Conclusions

The research presented in this paper deals with a two-level abstract indexing and storage scheme in the context of case-based plan recognition. World states representing planning situations are used as the indices for the indexing and retrieval of previously observed plans. Plan library is built incrementally from observations, with an indexing scheme that utilizes abstraction of concrete world states into non-negative integer vector representation. Such an indexing scheme focuses the recognizer to consider only a subset of the most relevant choices at retrieval time. Experimental results indicate significant improvements in prediction accuracy with the abstract indexing scheme. Further research will be concerned with replacing the random prediction choices with better ones and will utilize the argument substitutions for the next action predictions.

## Acknowledgments

This paper is supported by the Dayton Area Graduate Studies Institute (DAGSI) under grant #HE-WSU-99-09 (ABMIC) and by a grant from the Information Technology Research Institute (ITRI). I would also like to thank Michael Cox and Janette Giuseffi for their suggestions and support, as well an anonymous reviewer for valuable comments.

## References

Aho, A., Hopcroft, J., & Ullman, J. (1974). *The design and analysis of computer algorithms*. Reading, MA: Addison-Wesley.

- Allen, J. F., & Perrault, C. R. (1980). Analyzing intention in dialogues. *Artificial intelligence*, 15(3), 143-178.
- Amarel, S. (1968). On representations of problems of reasoning about actions. In Michie (Ed.), *Machine Intelligence 3* (pp. 131-171). Evanston, IL: Edinburgh University Press.
- Bares, M., Canamero, D., Delannoy, J. F., & Kodratoff, Y. (1994). XPlans: Case-based reasoning for plan recognition. *Applied artificial intelligence* 8, 617-643.
- Bodlaender, H. L. (1990). Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. *J. Algorithms*, 11, 631-643.
- Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M., & Wang, X. (1992). *PRODIGY 4.0: The Manual and Tutorial* (Tech. Rep. No. CMU-CS-92-150). Carnegie Mellon University, Department of Computer Science, Pittsburgh, PA.
- Colbourn, C. J., & Booth, K. S. (1981). Linear-time automorphism algorithms for trees, interval graphs, and planar graphs. *SIAM J. Comput.*, 10, 203-225.
- Fink, E. (1999). Automatic representation changes in problem solving (Tech. Rep. No. CMU-CS-99-150). Doctoral thesis, Carnegie Mellon University, Department of Computer Science, Pittsburgh, PA.
- Kautz, H. (1991). A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin, & J. Tenenber, *Reasoning about plans*. San Francisco: Morgan Kaufmann.
- Kerkez, B. (2001) *Incremental Case-based Keyhole Plan Recognition*. Technical Report, WSU-CS-01-01, Department of Computer Science and Engineering, Wright State University.
- Kerkez, B., & Cox, M. (2001). Case-based plan recognition using state indices, In D. W. Aha, I. Watson, & Q. Yang (Eds.), *Case-based reasoning research and development: Proceedings of 4<sup>th</sup> international conference on case-based reasoning* (pp. 227-242). Vancouver, Canada: Springer-Verlag.
- Kerkez, B., & Cox, M. (2002). Incremental Plan Recognition with Incomplete Plan Libraries, *Unpublished*.
- Kolodner, J. L. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.
- Lesh, N., & Etzioni, O. (1996). Scaling up goal recognition. In *Proceedings of the fifth internat. conference on principles of knowledge representation and reasoning* (pp 178-189).
- Luks, E. M. (1982). Isomorphism of graphs of bounded valence can be tested in polynomial time. *JCSSI*, 25, 42-65.
- Pollack, M. E. (1986). *Inferring domain plans in question-answering*. Doctoral thesis, University of Pennsylvania, Department of Computer Science, Pittsburgh, PA.
- Riesbeck, C. K., & Schank, R. C. (Eds.). (1989). *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Veloso, M. (1994). *Planning and learning by analogical reasoning*. Springer-Verlag.
- Wilensky, R. (1980). *Meta-Plannin*. Proceedings of AAAI80, Stanford, CA, pp. 334-336.