

CASE-BASED PLAN RECOGNITION WITH NOVEL INPUT

M. T. Cox* and B. Kerkez**

Abstract

Our research investigates a case-based approach to plan recognition using incomplete incrementally-learned plan libraries. To learn plan libraries, one must be able to process novel input. Retrieval based on similarities among concrete planning situations rather than among planning actions enables recognition despite the occurrence of newly observed planning actions and states. In addition we explore the benefits of predictions using a measure that we call *abstract similarity*. Abstract similarity is used when a concrete state maps to no known abstract state. Instead a search is performed for nearby abstract states based on a nearest neighbor technique. Such a retrieval scheme enables accurate prediction in light of extremely novel observed situations. The properties of retrieval in abstract state-spaces are investigated in three standard planning domains. We first determine optimal radii to use that determines a spherical sub-hyper-space that limits the search. Then experimental results show that significant improvements in the recognition process is obtained using abstract similarity.

Key Words

Case-base reasoning, plan recognition, nearest neighbor algorithms, abstract similarity.

* Intelligent Distributed Computing, BBN Technologies, Cambridge, MA 02138; mcox@bbn.com

** Department of Mathematics and Computer Science, Ashland University, Ashland, OH 44805; bkerkez@ashland.edu

1. Introduction

Plan recognition is intent inference from current observed behavior, given a series of old observations. Traditional plan recognition algorithms (e.g., [1,2,3,4]) all assume a complete plan library containing all possible plans, so performance under novel situations is not an issue by definition. Yet many problems exist for this approach including the burden imposed upon the knowledge engineer to enumerate all possible plans of large domains and the fact that many theoretically possible plans are extraneous [4]. Our research explores a case-based approach to plan recognition using incrementally learned plan libraries (i.e., case libraries) instead. Starting with an empty library, the plan recognition process must necessarily understand novel input to form the basis of understanding future input. This research incorporates case-based reasoning (CBR) and state abstraction with plan recognition in a novel manner to handle unknown situations. Using abstraction a case-based recognizer (i.e., observing agent) can predict the behavior of an observed agent, because it uses the current situation to map to an abstract state that indexes all old cases sharing this abstract relationship. It then uses a selected old case for prediction and interpretation. This provides robustness in the face of novel actions and novel states. However this ability depends upon the past plan¹ having a state that maps to the same abstract condition to which the currently observed state maps. *When no past plan exists in the case library that contains such a state*, the approach is unable to make an informed prediction as to the next action the observed agent will perform. Although the details of case creation is shown elsewhere (see [5] and [6]), in this paper we report a new approach based upon nearest-neighbor techniques that allows a past case to be found under such circumstances.

Rather than solve every problem from scratch, case-based reasoning [7,8,9] uses past experience in

1. Observed agents are assumed to be acting in service of their goals, and therefore, the actions they perform constitute a plan to achieve such goals. Cases are simply records of past plans used by the observed agent.

the form of previously solved problems to solve new problems that share similar situations. The classical CBR process is to retrieve an old but similar case, adapt the old solution to fit the new circumstances, apply and evaluate the solution, and then store the new solution if sufficiently different from known solutions. Likewise in our approach to plan recognition, indexing and retrieval of past plans are based on recalling the previously observed situations (i.e., states) that are the most similar to the current situation at hand, and then using past similar situations to determine the observed agent's current course of action.

In contrast traditional recognition systems search for matching planning actions in a set of stored plans. These plans are simply sequences of actions represented as operators such as `stack(BlockA BlockB)` whose executions will transform an initial state of a given problem into a goal state. To use the case-based approach, however, the plans must contain the intermediate states between actions to provide matching situations. Therefore we represent plans as a sequence of *action-state pairs* [10] that begins with the null operator paired with the initial state and ends with the final operator linked with the goal state. As a consequence, our approach enables the observing agent to reason about novel planning actions. After executing a novel action, the observed agent may be in a world state that is similar to some previously observed state. The observer can then refer to the agent's previous behavior in similar past situations in order to infer its future intentions. Traditional plan recognition systems will have no match to novel actions, and thus they will not be able to continue.

Although the planning situations represented by the states of the environment are useful as indices for recognition, their practical use is limited due to potentially large state-spaces in complex domains. We cope with this complexity by utilizing a multiple-level indexing scheme that enables efficient retrieval of past plans. World states, represented as collections of ground literals, can be abstracted into non-negative

integer vectors by counting the number of occurrences of type-generalized predicates in a single state and by placing the counted sum in the appropriate dimension. Fig. 1 shows an example state in the block-world planning domain. $S_{1,1}$ consists of four blocks whereby blocks B, A, and C are on a table and block D sits upon block C.

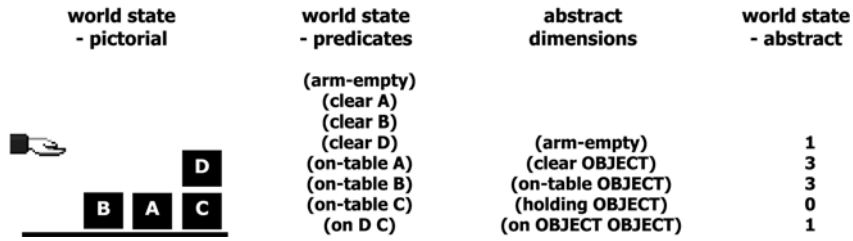


Figure 1. An example of a blocksworld domain state, $s_{1,1}$, and its different representations.

Now consider a concrete plan, P_1 , to build a second column with B on A. The goal can be achieved by picking up B and placing it on A. In general, a concrete plan, P_i , is an ordered sequence of action state pairs, $(a_{i,j}, s_{i,j})$, where i ranges from 1 to the total number of plans in the case base, and j ranges from 1 to the number of steps in P_i . In the specific plan P_1 , $a_{1,1}$ is the null action, and the initial state is $s_{1,1}$ from fig. 1. The first action, $a_{1,2}$, is `pick-up(B)` and leads to the concrete state, $s_{1,2}$, with the predicate `(holding B)` true. Because the operator also deletes the states `(on-table B)`, `(clear B)` and `(arm-empty)`, the representation of this new abstract state, as_2 , is $[0\ 2\ 2\ 1\ 1]$. Then executing $a_{1,3} = \text{stack}(B,A)$ achieves the goal state, $s_{1,3}$, whose abstract representation is the state $as_3 = [1\ 2\ 2\ 0\ 2]$. Therefore the representations for the concrete and abstract plans are as follows.

$$P_1 = (\text{null}, s_{1,1}), (\text{pickup}(B), s_{1,2}), (\text{stack}(B,A), s_{1,3})$$

$$AP_1 = (\text{null}, [1\ 3\ 3\ 0\ 1]), (\text{pick-up}(\text{OBJECT}), [0\ 2\ 2\ 1\ 1]), (\text{stack}(\text{OBJECT}, \text{OBJECT}), [1\ 2\ 2\ 0\ 2])$$

Abstract states are employed as the first level of indexing, because many concrete states may share the identical abstract representation [10]. Given this property an observing agent can find old concrete states (and thus old plans) that match at the abstract level even though the current concrete state is novel. Fig. 2 illustrates the indexing structures used for the plan storage and retrieval processes. The abstract state-space contains the abstracted states, as_k , where k ranges from 1 to the number of abstract states, and the observed plans, P_i , in their abstract and concrete representations. In this figure, we can see the example plan, P_1 , in the case base. Abstract states point to structures called *bins* that contain concrete states ($s_{i,j}$) sharing an identical abstract representation.

Serving as a second level of indexing, a *pseudo-isomorphic equivalence relation* [6] provides a natural partitioning of bin members into disjoint equivalence classes. In brief this relation is obtained by transforming the concrete states into a directed hyper-graph representation such that each no-argument predicate (i.e., proposition) and each object in the state form the vertices. Reflexive arcs correspond to unary predicates, edges between two vertices correspond to binary predicates, and hyper-edges correspond to all other predicates. For each vertex, the equivalence-relation algorithm generates a set of connection strings [6] for all incoming and outgoing edges. For the example state of fig. 1, the vertex representing block C has two such strings. Because of the edge representing (on-table C), the algorithm produces “on-table1.” Due to the edge from vertex C to D labelled on, the algorithm produces “on2.” The numeral 1 represents an outgoing edge, and 2 represents an incoming edge. The complete connection string is then “on2on-table1.” Finally strings for each graph are sorted and then compared with each other. If all strings match, the pseudo-isomorphism relation exists. The algorithm guarantees that if two string sets do not match, then their corresponding states are not truly isomorphic, thus providing a consistent partitioning for use as an index. If they do match, they may or may not be isomorphic. However the

algorithm is linear in the sum of the number of vertices and edges, whereas computing actual isomorphism is suspected to be in the complexity class NPI .

Now given the currently observed planning situation, the observing agent transforms it into an abstract representation, locates the matched bin and appropriate equivalence class, and retrieves all past situations from the matched equivalence class. The observer will then retrieve all previous cases containing the matched past situations via the state pointers into the plan library (see fig. 2). These past cases may then be utilized to infer the planner's current intent. Selection of a particular case to use for inference is determined randomly or by frequency (the selection criterion used in this paper). Although many heuristics might be used to improve upon such knowledge poor criteria including weighting particular vector dimensions, we leave this for future research.

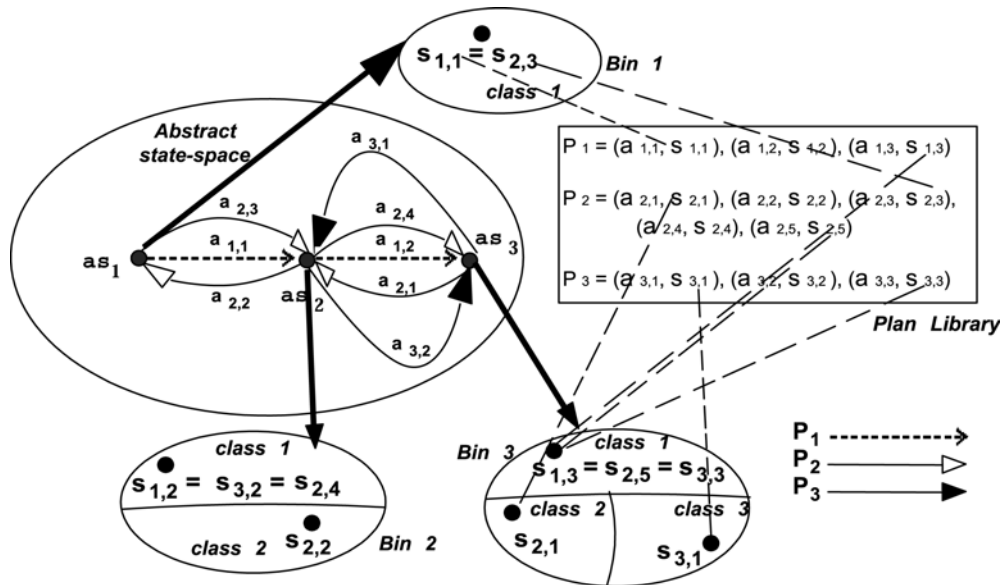


Figure 2. Multi-level indexing structures. Abstract states (as_k) point to bins with disjoint equivalence classes, containing concrete past situations ($s_{i,j}$) that point to the past plans (P_i) in which they are contained through state pointers (dashed lines into the library).

We distinguish between two types of intent. *Local intent* refers to what the agent intends to do next (i.e., its next action) in its plan; whereas *global intent* refers to what overall state of affairs (i.e., its final goal state) the agent wants to achieve with the plan. These two types of intent correspond to what Grosz and colleagues called intention-to and intention-that [11]. Past experimental results show the effectiveness of the retrieval scheme based on situational reminding and multiple indexing levels in recognition with incomplete plan libraries [6]. In this paper we explore plan recognition performance using both types of intent prediction as measurement criteria.

The indexing scheme described above enables the observing agent to reason in light of novel planning actions and states. Upon execution of a novel action, the observed planning agent may reach a situation that is the same as or very similar to some previously observed situation. Past plans containing the situation state may then be used to predict the planner’s intent (i.e., next action or final goals), although the currently observed action was a completely new one. The indexing scheme will likewise enable recognition and prediction when the currently observed situation is novel, provided that its abstract representation can be found in the abstract state-space (i.e., maps to an existing bin). As a local intent example, when a newly observed state has the blocksworld hand holding block A with B on the table and has D stacked on C (which is similar to the intermediate state of the example above), a case-based plan recognition system should predict that A will be placed on block B (by appropriately adapting $a_{1,3}$ of plan P_1). However, retrieval failure occurs when a completely novel abstract state is observed. For example the recognizer may observe all four blocks on the table (which maps to $[1\ 4\ 4\ 0\ 0]$ and therefore no existing bin). This failure reduces performance, because the system will be unable to make a prediction. It could however make a prediction (e.g., that a pick up action may occur next) if it could recognize that the new state is “close” to another (e.g., the initial state’s abstract representation of $as_1 = [1\ 3\ 3\ 0\ 1]^2$ from fig. 1).

Using such closeness judgements, in terms of what we call abstract similarity, to improve the prediction accuracy of plan recognition represents the major contribution of this paper.

In [9] and [10], detailed explanations exist for the representations and algorithms that constitute the core recognition processes outlined here by the introduction. In the remainder of this paper, we describe an extension of the retrieval scheme that enables recognition in light of novel abstract situations. The system utilizes retrieval based on the similarity in the abstract state-space. Because the abstract states are non-negative integer vectors, a similarity metric using a nearest neighbor algorithm is a natural choice. The next section discusses situation similarity-assessment at the abstract level in more detail, while the following section illustrates experimental evaluations of this technique on three planning domains.

2. Abstraction and Similarity

In general two types of similarity exist in our framework. When two different concrete states share an abstract representation, we say that they have *concrete similarity*. Thus when two concrete states appear in separate bins, they are concretely dissimilar. However, if the two bins are within close proximity to each other relative to other bins, we say they share *abstract similarity*. The criterion that determines proximity is a nearest neighbor function in the abstract search space.³ As mentioned in the previous section,

2. Euclidean distance between two points, $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$, is measured as $D(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$,

so the distance to the new observation is $\sqrt{3}$ as opposed to $\sqrt{11}$ and $\sqrt{12}$ for as_2 and as_3 respectively.

3. Although many methods exist in the machine learning and CBR communities that do a more efficient job at estimating similarity, we use nearest neighbor because of its conceptual simplicity. Alternatives such as fuzzifying the distance rather than discretizing it would allow more bins to contribute to plan recognition, but here we wish to merely establish the efficacy and appropriateness of abstract similarity within our framework rather than to optimize its application.

each bin is located as a point in an abstract space represented by its abstract feature vector. In this manner we can set a sphere within the abstract space at a given radius and define abstractly similar points to be within this sub-space. To try to use a nearest neighbor approach directly in the concrete space is prohibitively expensive because of the large vector size for concrete points, but in the abstract space we need only calculate distances for all points in the sphere rather than all points in the abstract space.

Upon observation of an execution of some action, the action along with the current state reached after the action application are received by the observer. The observer then searches for a similar past plan with which to interpret the observation. Interpretation consists of being able to predict the global and local intentions of the observed. For each of these types of intentions, the system makes two types of predictions. An *abstract prediction* provides the intention at the abstract space (e.g., pickup(BLOCK)), whereas a *concrete prediction* provides a fully instantiated intention with ground literals (e.g., pickup(BlockC)). Prediction is possible by adapting the next intention of the past plan, and adaptation is accomplished by changing the object arguments of the past actions or state predicates to objects mapped into the current state. The subsequent observation provides the action with which to evaluate local intention (the predicted action), and the last observation of the input sequence provides the state with which to evaluate global intention (the predicted final or goal state). A crucial part of this process is finding a suitably similar past sequence of observations.

To find this past sequence, the currently observed state is linearly transformed into its abstract representation, and the observer attempts to locate a bin that contains the abstract point. This is a search for concrete similarity. In case of a successful match at this level, the retrieval process focuses on the found bin. The observer subsequently searches the bin to find an equivalence class that matches the state. If the

current state is a member of an existing equivalence class, past cases containing situations from the matched equivalence class are retrieved from the library via the state pointers (see fig. 2). We call this a match at the *class level*. If the current state is not a member, a new equivalence class is created in the matched bin, and the newly observed state is stored in the created equivalence class. We call this a match at the *bin level*. After such matches, the observing agent retrieves past cases pointed to by all of the concrete situations in the matched bin. Although an equivalence class-level match is more focused than a match at only the bin level, the latter type of a match is still very useful. The abstract indexing scheme guarantees that all states in a single bin are concretely similar, because they all share an identical abstract representation.

To understand better the difference between these two types of matches, we return to the example using plan P_1 from the introduction. The final state of P_1 is the configuration where B is on A and D is on C and the abstract representation is [1 2 2 0 2]. To state this in other terms, the concrete state is {(arm-empty)(clear B)(clear D)(on-table A)(on-table C)(on B A)(on D C)} which maps to one arm-empty, 2 clears, 2 on-table, no holding, and two on predicates. Now the state in which labels are simply rearranged, for example A might be on B instead of B on A, is clearly very similar. In fact it not only shares an abstract vector representation with the original state, but it also has the same physical structure in the world. The two graph representations of the states are isomorphic, so they are both in the same equivalence class. Now consider the same state except that B is on D rather than on A. This state has a single three block tower B-D-C and is clearly different structurally. But notice that it too maps to the vector [1 2 2 0 2]. Therefore it matches the goal state of P_1 at the bin level, but not at the class level. It exists in a separate equivalence class from either of the other two states but shares the same bin. In other words, all three states are concretely similar despite their differences both large and small.

The observer is not able to find matches, however, when the currently observed planning situation has a novel abstract representation. Match failure at the level of abstract states indicates presence of a newly observed abstract state and prevents the observer from finding any concretely similar past situations to guide the recognition process. Instead of attempting to find an exact match at the abstract state level, the observer will attempt to find other abstract states that are similar to the abstract representation of the currently observed planning situation. Because abstract states are non-negative integer vectors, a nearest-neighbor similarity metric can be used to locate similar states in the abstract state-space. Once such similar abstract states are found, the observer may be able to use past situations from similar bins to infer the planner’s current intent.

Fig. 3 illustrates retrieval based on abstract similarity. It shows a portion of an abstract state-space along with example indexing structures. Upon observing the current action A_{curr} and the resulting novel abstract state as_{new} , the observer creates another bin, Bin z, with a single equivalence class in which as_{new} is stored. By using the nearest-neighbor similarity metric, the observer then locates all abstract states whose distance from the newly observed abstract state is less than or equal to a specified radius. Fig. 3 shows an example in which two similar abstract states are found at distances $da1$ and $da2$ in the abstract state-space within the specified maximum distance of R . These states point to bins d and v that contain past situations at distances $d1$ and $d2$.⁴ All concrete states contained in these bins are then collected, and the most frequent state is chosen. A random case containing that state provides the guidance for local intent (i.e., next action) prediction.

4. In contrast to $da1$ and $da2$, it is impractical to actually compute $d1$ and $d2$, because the concrete hyperspace points would consist of bit vectors of length equal to the number of possible predicates in the concrete state space.

the current planning situation. Consequently, predictions based on the abstract similarity metric are less likely to be correct than predictions based on an exact match at the level of abstract states (i.e., based on concrete similarity). This is especially the case when the radius for the abstract similarity measure is large and the nearest neighbor scheme retrieves a large number of bins that fall within the specified maximum distance. Nevertheless, some of these predictions may be correct and therefore improve the overall prediction accuracy of the observer.

3. Empirical Results

To investigate the characteristics of the retrieval strategy based on abstract similarity, we investigated the performance of our plan recognition implementation with and without the similarity measure. We focused our evaluations on plans in the logistics (i.e., package delivery) domain [12] and the extended-STRIPS [13,14,15] robot domain. We examined two variations of the extended-STRIPS domain. In the ex-STRIPS variation we allowed problems having multiple top-level goals; whereas in ex-STRIPS-OG, all problems had but *one* top-level *goal* (hence the OG). We created a planning problem generator to produce large sets of random problems and then solved these problems to produce plans that constituted the observed agent's actions and thus the input for the observing agent. These plans are obtained from executions of the PRODIGY [13,16] state-space nonlinear planner on the problem sets. The execution cycle of the planner was modified to monitor and report the intermediate planning states along with the executed planning actions. The recognition process was executed on a large scale with about 50,000 observed planning steps for the logistics domain and approximately 11,000 and 17,000 steps for the two ex-STRIPS domains. See [5] for technical details concerning the problem generator and the modifications to PRODIGY.

Fig. 4 shows a logarithmic plot of the average number of past concrete states that are found in retrieved bins for each retrieval step based on a match in the abstract state-space. Although the implementation allows the user to choose between a Euclidian and a city-block (Manhattan) distance measure, these evaluations illustrate the nearest neighbor similarity metric with the Euclidian distance measure. The results obtained with the city-block distance measure are very similar to evaluations presented here. The x-axis depicts different radii that represent the maximum allowable distance value at which two abstract states are considered to be similar. Notice that because abstract feature vectors consist of non-negative integers, the figures in this subsection show evaluations that consider all possible distance intervals between $\sqrt{2}$ and 3 inclusively as well as several radii at larger distances (4 through 6, as indicated on the x-axis of the figures in this subsection).⁵

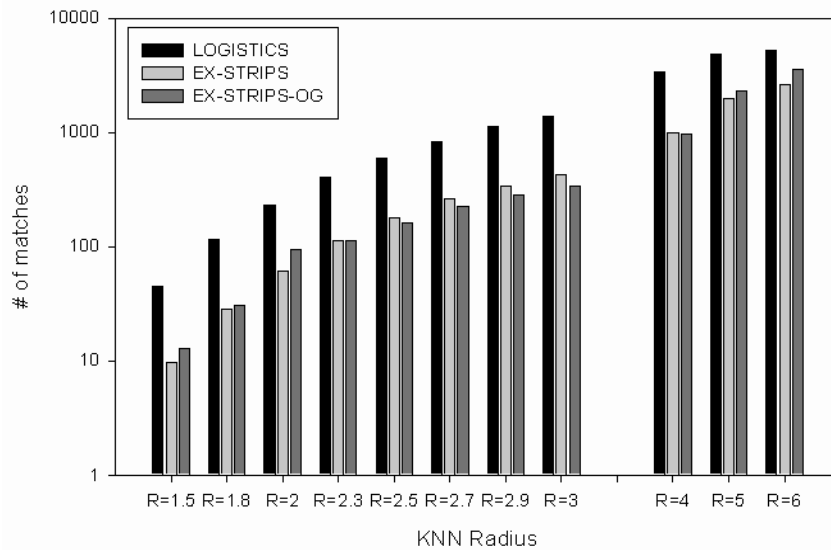


Figure 4. Logarithmic plot for average numbers of past concrete states from bin matches within a specified radius of the nearest neighbor based retrieval in the abstract state-space.

5. Radius 1 is not used, because for any N-vector, only 2N points will exist within the sphere. That is only one dimension can differ between the reference point and any other point within a ball with radius one, and the dimension can differ only by ± 1 .

Fig. 4 clearly shows that as the radius increases, so does the average number of past states found after retrieval in the abstract state-space. When the number of past concrete states within the sphere is large, the observer will have to consider a very large number of past plans. Therefore, this large size of matches may negatively impact the efficiency of the observing agent. The issue then is to determine experimentally what radius will produce the best increase in performance.

The overall benefits of retrieval based on abstract similarity can be observed in fig. 5 and fig. 6. They show the percentages of overall increase in local prediction accuracies at both levels of abstraction with respect to retrieval without abstract similarity. We evaluated the nearest neighbor scheme with the Euclidean distance measure. Evaluations span across several radii for a sphere around a domain's feature vector representing the maximum allowable distance at which similar abstract states may be located. For the logistics domain, a radius of 1.8 produces the best performance for both abstract and concrete predictions. In both of the extended STRIPS domains, a radius of 1.5 proves best.

Figs. 7 and 8 compare the improvements in global prediction accuracies for all three evaluation problem sets at the level of abstract and concrete goals, respectively. Note that the values for fig. 7 at radii 2.3 and 2.5 are equal to 0.006112096, although they do not appear visible in the figure. The improvements in global prediction accuracies for these problem sets are similar to the improvements in local prediction accuracies shown in the previous paragraph. These figures also show that the improvements in global prediction accuracies are much greater at the concrete than at the abstract level. Because concrete goal predictions are much more informed than abstract goal predictions, the results shown in these figures are very encouraging. Furthermore, a radius of 1.5 generates the highest improvement in performance with abstract global predictions. But with respect to concrete global predictions, 1.5 is best for only ex-

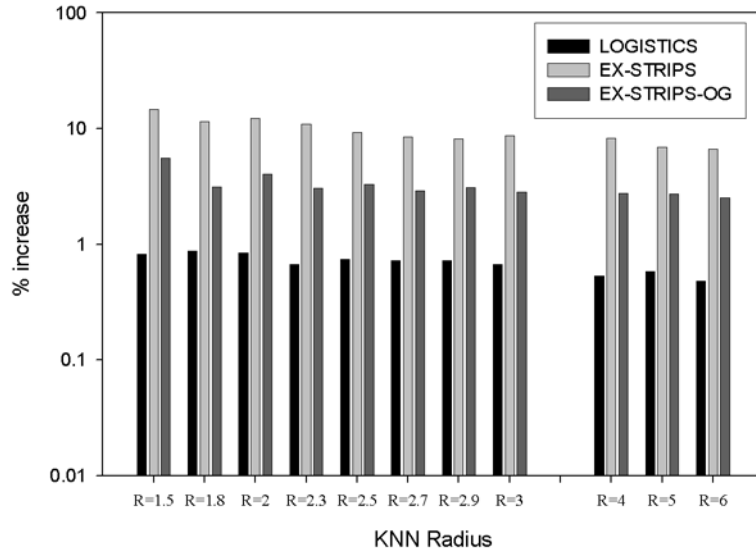


Figure 5. Logarithmic plot showing a comparison between increases in *abstract local prediction* accuracies over the strategy without abstract similarity for all evaluation problem sets.

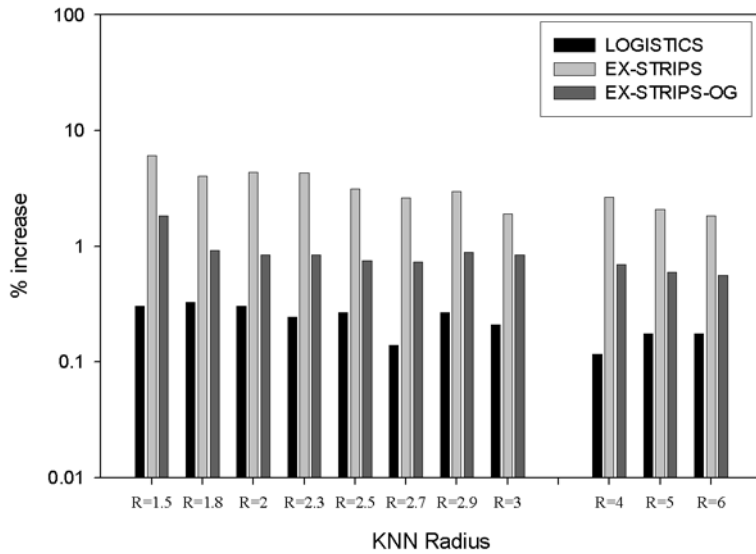


Figure 6. Logarithmic plot showing a comparison between increases in *concrete local prediction* accuracies over the strategy without abstract similarity for all evaluation problem sets.

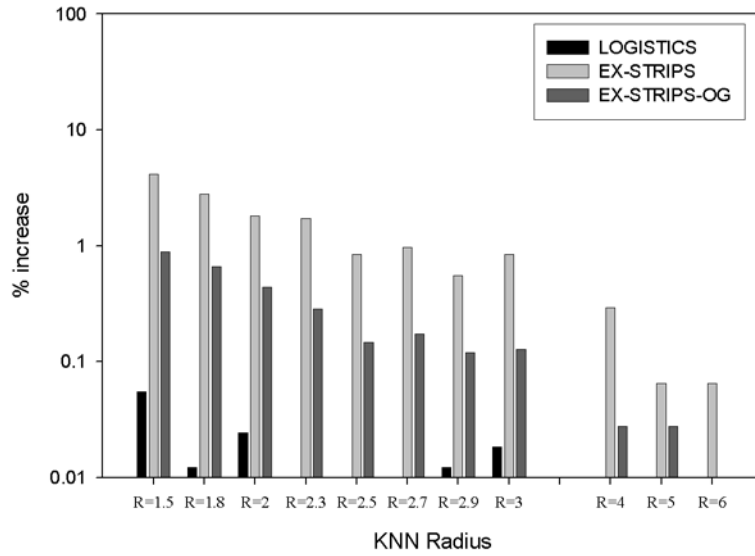


Figure 7. Logarithmic plot showing a comparison between increases in *abstract global prediction* accuracies over the strategy without abstract similarity for all evaluation problem sets.

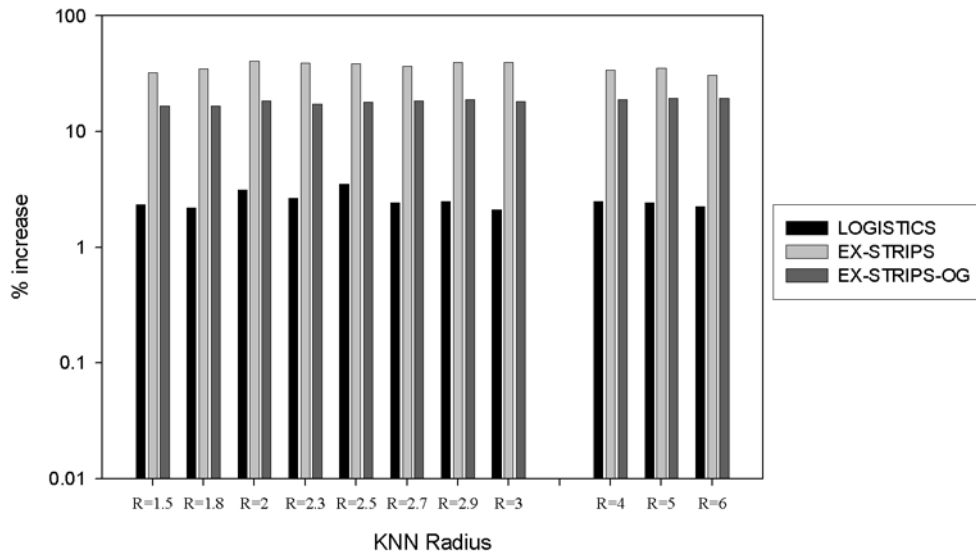


Figure 8. Logarithmic plot showing a comparison between increases in *concrete global prediction* accuracies over the strategy without abstract similarity for all evaluation problem sets.

STRIPS-OG. For both logistics and ex-STRIPS, a radius of 2 is better.

Finally by comparing figs. 5 and 6 with figs. 7 and 8 we see a noticeable trend in the data. Local predictions benefit more from our method at the abstract prediction level than at the concrete prediction level. Whereas with global predictions the trends are reversed. The improvements in accuracy occur highest at the concrete prediction level. One reason for this difference is that global goal predictions must specify the entire state, not just a predicate expression. Thus the space of possibilities is huge. For global predictions the actual accuracy ratings are small, and although the absolute change of performance is roughly the same for both concrete and abstract global prediction, the relative difference is far greater at the concrete level (see Table 1, below).

Figs. 9, 10, and 11 show comparisons of local prediction accuracies with and without retrieval based on the abstract similarity. The figures depict cumulative accuracies throughout the incremental establishment of the case library beginning with no past plans and provide base-line performance numbers to compare against. They show both abstract and concrete predictions using the best performing nearest-neighbor radii. That is, the logistics domain uses a radius of 1.8, whereas both extended STRIPS domain variations use 1.5. We can see that in each case utilization of abstract similarity results in increased prediction accuracies.

The ex-STRIPS problem set shows largest improvements at the level of both abstract and concrete action predictions. The smallest (although statistically significant given the large number of observations) improvements in local prediction accuracies occur in the logistics problem set (fig. 9). The logistics problem set is characterized by a relatively small abstract state-space and a steady-state recognition behavior

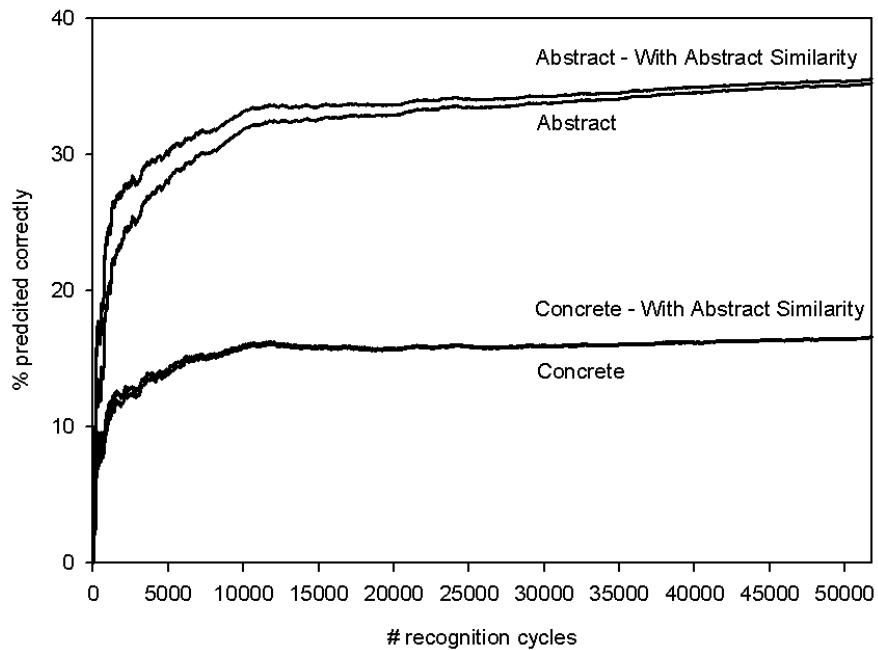


Figure 9. Comparison of local prediction accuracies with and without abstract similarity for the logistics problem set. Base-lines: abstract = 11.72; concrete = 1.72.

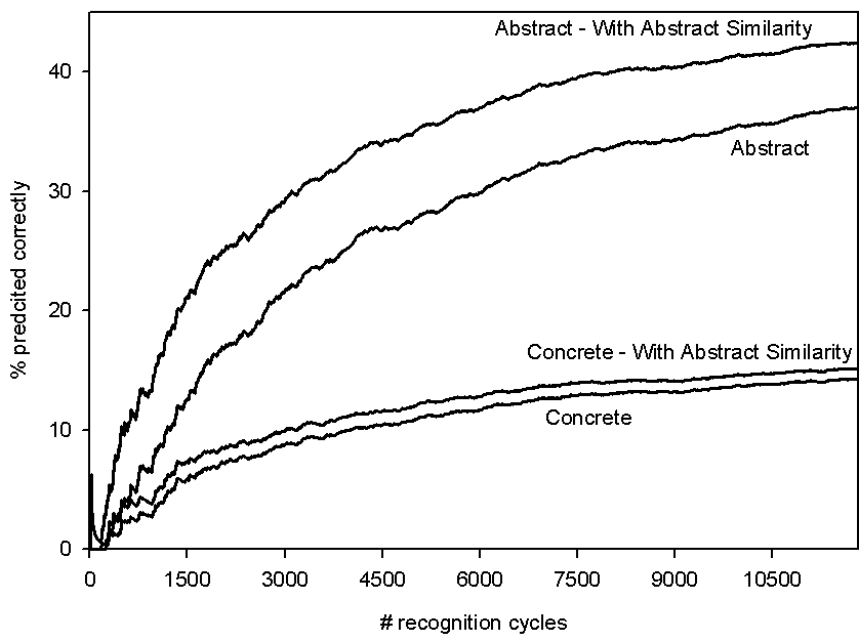


Figure 10. Comparison of local prediction accuracies with and without abstract similarity for the *ex-STRIPS* problem set. Base-lines: abstract = 11.98; concrete = 1.02.

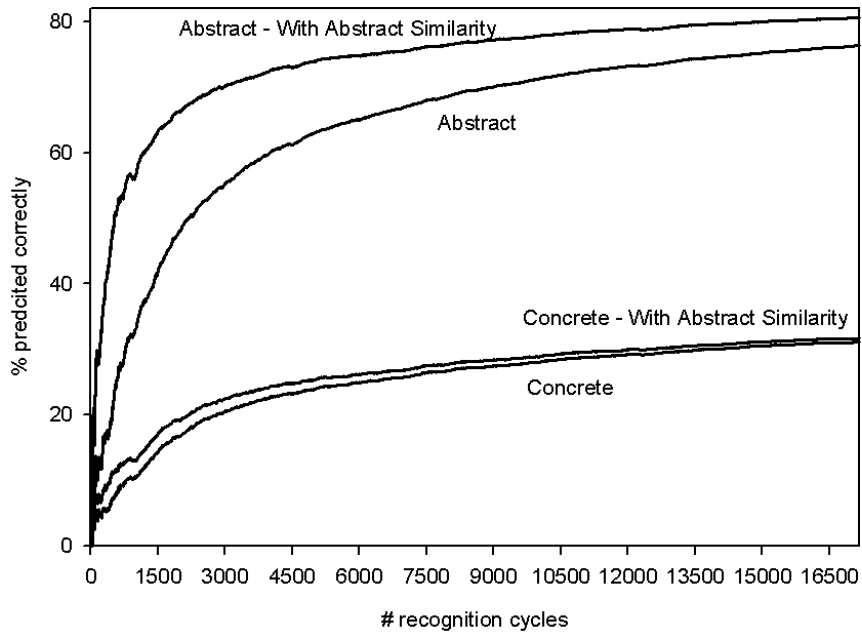


Figure 11. Comparison of local prediction accuracies with and without abstract similarity for the *ex-STRIPS-OG* problem. Base-lines: abstract = 20.21; concrete = 2.41.

in the long run. Therefore, the observer rarely has the need to search the abstract state-space for matches, because novel abstract situations are seldom observed as recognition reaches the steady state. Contrastingly, the *ex-STRIPS* problem set has the largest rate of creation of new bins, so the observer often encounters a situation with a novel abstract representation. Thus a number of retrieval attempts in the abstract state-space in the *ex-STRIPS* problem set is much greater than the corresponding number in the logistics problem set. Because the observer attempts more retrievals in the abstract state-space in the *ex-STRIPS* problem set, it also has more chances to make correct predictions in this problem set than in the other two problem sets.

The plots for global predictions (not shown here) are very similar in shape compared to the plots of local predictions above. The heights of the curves are different however. The final accuracies after all

observations (i.e., those at the extreme value along the x-axis) for each domain are as shown in Table 1:

Table 1
Final Global Prediction Accuracies

	Logistics	ex-STRIPS	ex-STRIPS-OG
Abstract prediction; abstract similarity ^a	31.62	27.14	64.20
Abstract prediction; concrete similarity	31.61	26.06	63.64
Concrete prediction; abstract similarity	2.54	7.34	9.70
Concrete prediction; concrete similarity	2.48	5.54	8.32

a. This first row includes performance when abstract similarity is used in those cases where concrete similarity fails. Thus the term “abstract similarity” means using both abstract and concrete similarity as opposed to concrete similarity alone.

The three domains we used for evaluation purposes have quite different characteristics. The logistics domain is characterized by a smaller abstract state-space than ex-STRIPS, because it consistently generates a fewer number of bins for the three domain sets. This results in a smaller number of retrievals based on the nearest neighbor similarity metric in the logistics domain. While the recognition process reaches a steady-state behavior in the logistics domain, it fails to do so in the ex-STRIPS domain, due to a high rate of observation of new abstract states even after 11,000 observed planning steps. A larger number of observed planning steps in the logistics domain, shown in the left column of fig. 4, is also an important factor for achieving the steady state behavior in this domain.

4. Conclusions

In this paper we presented an extension to the case-based plan recognition scheme with incomplete plan libraries. The CBR methodology employs a retrieval mechanism based on similarity among planning situations represented by the states of the planner’s environment. This is opposed to traditional plan recog-

dition algorithms that employ matching of agent actions. Our extension utilizes a nearest neighbor similarity metric that enables retrieval in an abstract state-space. This retrieval using abstract similarity occurs when the current planning situation being observed has a novel abstract representation. Although evaluations show that the retrieval based on partial matches in the abstract state-space is effective, the degree of effectiveness depends on the state space characteristics of a given domain and the radius of the subspace defining similarity. The results presented in this paper indicate that retrieval in the abstract space is more effective in planning domains with large abstract state-spaces where steady-state recognition behavior is not reached quickly. Through empirical investigation, we showed that radii between 1.5 and 2.5 perform best across domains and prediction types (i.e., concrete versus abstract and local versus global). Our prediction is that this range of numbers will remain optimal given any domain. Future research efforts will investigate nearest-neighbor case-based retrieval in planning domains with different state-space characteristics to confirm this.

Many potential applications of our representations and plan recognition techniques exist. For example Fagan and Cunningham [17] have developed an intelligent version of the game Space Invaders based directly on our research. These methods enable their system to predict the actions of a user's opponents by learning a case-based plan library and retrieving old plans. More generally if a given domain can be represented within a state-space using traditional planning operators, then case-based plan recognition can lead to predictable behavior with a tractable implementation. The scalability of our approach is certainly established, given that we use more than 60,000 observations in the logistics domain alone. Another interesting application we foresee is found in mixed-initiative intelligent systems (see for example [18]) that integrate the user as an active participant in decision making. Here events produced by the human user act as actions that change the state of the interface. Thus tutoring and intelligent help systems

could better model the user and know when to intervene to provide help or information.

The current implementation of case-based plan recognition adds new bins, equivalence classes, and plans to the case-base throughout the recognition process. However, this limits performance, because bins will eventually become saturated so that the complexity of search is no longer controlled by our indexing techniques. We believe that once the prediction rate achieves asymptote, the incremental development of the case base (i.e., learning) should stop. An interesting approach to determining when a case-base is sufficiently populated is to halt the learning when the system rarely has to use abstract similarity. Computational experiments in the future may reveal a sufficient threshold to use in arbitrary domains.

Acknowledgment

This research was supported by AFOSR under grant #F49620-99-1-0244 and by DAGSI/AFRL. Further support was received from the Information Technology Research Institute and from the Ohio Board of Regents. We also thank the five anonymous reviewers. In particular one reviewer suggested the alternative of fuzzification of distances which represents an interesting alternative.

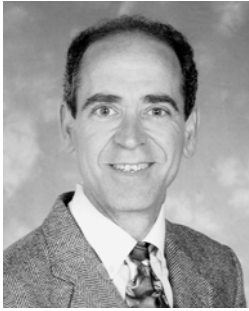
References

- [1] M. Bauer, Machine learning for user modeling and plan recognition, in V. Moustakis, & J. Herrmann (Eds.), *Proceedings of the ICML'96 Workshop on machine learning meets human computer interaction*, 1996, 5-16.
- [2] E. Charniak, & R. Goldman, A bayesian model of plan recognition, *Artificial Intelligence* 64, 1993, 53-79.

- [3] H. Kautz, *A formal theory of plan recognition*, doctoral diss., University of Rochester, Computer Science Dept., Rochester, NY, 1987.
- [4] N. Lesh, & O. Etzioni, Scaling up goal recognition, *Proc. of the Fifth International Conf. on Principles of Knowledge Representation and Reasoning*, 1996, 178-189.
- [5] B. Kerkez, *Incremental case-based plan recognition with incomplete plan libraries*, doctoral diss., Wright State University, Department of Computer Science and Engineering, Dayton, OH, 2003.
- [6] B. Kerkez, & M.T. Cox, Incremental case-based plan recognition with local predictions, *International Journal on Artificial Intelligence Tools: Architectures, languages, algorithms*, 12(4), 2003, 413-463.
- [7] J.L. Kolodner, *Case-based reasoning* (San Mateo, CA: Morgan Kaufmann, 1993).
- [8] C.K. Riesbeck, & R.C. Schank (Eds.), *Inside case-based reasoning* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1989).
- [9] Leake, D. (Ed.), *Case-based reasoning: Experiences, lessons, & future directions* (Menlo Park, CA: AAAI Press / MIT Press). 1996.
- [10] B. Kerkez, & M.T. Cox, Case-based plan recognition using state indices, D.W. Aha & I. Watson (Eds.), *Case-Based Reasoning Research and Development: Proc. of 4th International Conf. on Case-Based Reasoning* (Berlin: Springer, 2001) 227-242.
- [11] B. Grosz, L. Hunsberger, & S. Kraus, Planning and acting together, *AI Magazine*, 20(4), 1999, 23-34.
- [12] M. Veloso, *Planning and learning by analogical reasoning* (Berlin: Springer, 1994).
- [13] J.G. Carbonell, J. Blythe, O. Etzioni, Y. Gil, R. Joseph, D. Kahn, C. Knoblock, S. Minton, A. Perez,

- S. Reilly, M. Veloso, & X. Wang, *PRODIGY 4.0: The manual and tutorial* (Tech. Rep. No. CMU-CS-92-150). Carnegie Mellon University, Department of Computer Science, Pittsburgh, PA., 1992.
- [14] R. E. Fikes, & N. J. Nilsson, STRIPS: A new approach to theorem proving in problem solving, *Artificial Intelligence*, 2, 1971, 189-208.
- [15] M. Veloso, & J.G. Carbonell, Case-based reasoning in PRODIGY. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning IV: A multistrategy approach* (San Francisco: Morgan Kaufmann, 1994) 523-548.
- [16] M. Veloso, J.G. Carbonell, A. Perez, D. Borrajo, E. Fink, & J. Blythe, Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical and Experimental Artificial Intelligence*, 7(1), 1995, 81-120.
- [17] M. Fagan, & P. Cunningham, Case-based plan recognition in computer games, *Proc. of the 5th International Conf. on Case-Based Reasoning* (Berlin: Springer, 2003).
- [18] M. T. Cox (Ed.), *Proceedings of the 1999 AAAI-99 Workshop on Mixed-Initiative Intelligence* (Menlo Park, CA: AAAI Press, 1999).

Biographies



Michael T. Cox is a senior scientist in the Intelligent Distributing Computing Department of BBN Technologies, Cambridge, MA. Previous to this position he was an assistant professor in the Department of Computer Science & Engineering at Wright State University, Dayton, Ohio, where he was the director of Wright State's Collaboration and Cognition Laboratory. He received his Ph.D. in Computer Science from the Georgia Institute of Technology, Atlanta, in 1996 and his undergraduate from the same in 1986. From 1996 to 1998, he was a postdoctoral fellow in the Computer Science Department at Carnegie Mellon University in Pittsburgh working on the PRODIGY project. His research interests include case-based reasoning, collaborative mixed-initiative planning, intelligent agents, understanding (situation assessment), introspection, and learning. More specifically, he is interested in how goals interact with and influence these broader cognitive processes. His approach to research follows both artificial intelligence and cognitive science directions.



Boris Kerkez is an assistant professor in the Mathematics and Computer Science Department at Ashland University in Ashland, Ohio. He received his Ph.D. in Computer Science and Engineering from Wright State University in 2003, and his M.S. in Mathematics and Statistics from Miami University in Oxford, Ohio in 1998, where he also completed his undergraduate studies in Systems Analysis. His research interests include machine learning, case-based reasoning, planning and plan recognition, human-machine interaction, Steiner Trees, Pancake Networks, and neural network applications.