Using Heuristic Search to Retrieve Cases that Support Arguments

Edwina L. Rissland, David B. Skalak and M. Timur Friedman
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{rissland, skalak, friedman}@cs.umass.edu

Abstract[1]

We discuss the use of heuristic search in a graph of cases and other domain knowledge to supply support for an argument. We contend that for legal argument, a gap exists between the vocabulary of constraints upon arguments and the indexing vocabulary for cases and other domain resource knowledge. We suggest best-first search of a graph of cases and other domain knowledge as a means to partially close this gap. In particular, search of the *case-domain graph* is driven by one of three evaluation functions that use different levels of abstraction to mediate between the indexing vocabulary and the task vocabulary. We describe the **BankXX** system, which uses heuristic search to retrieve cases and other domain knowledge in support of an argument.

Introduction

Indexing and Heuristic Search

Case-based reasoning (CBR) is often used in service of a task such as teaching, planning, design or argumentation. Part of the challenge in building CBR systems is to create a bridge between the superficial level at which cases can be input and the constraints of the ultimate task. A variety of useful techniques have been used to good advantage to provide case indexes that support the task at hand, including, among others, influence graphs [Sycara & Navinchandra, 1991], explanation-based generalization [Barletta & Mark, 1988], thematic abstractions [Owens, 1988], and the declarative reification of task constraints [Ashley & Aleven, 1991]. We focus on the task of creating legal arguments, where the vocabulary of the constraints on an emerging argument is different from the indexing vocabulary immediately available for case retrieval. We propose best-first search of a case base as one tool for case retrieval in support of an application task such as legal argument.

To take an extreme (but real) example of vocabulary mis-match between a task and the immediately available cases and indices, suppose that the cases are full-text legal opinions and Boolean combinations of keywords are the only indices available. Further suppose the requirement of the argument is to supply a case that uses the opponent's best theory, so that one can distinguish the case from the current problem and thereby discredit application of the opponent's theory. The constraints of this task cannot be readily expressed in terms of the available indices: there is a mis-match between the indexing and the task vocabularies[2]. Barring revision of the indexing vocabulary or a re-conceptualization of the domain, some search of the information resources may improve case retrieval.

Indexing and search present two extremes for retrieval. At one extreme, indices may function as database retrieval keys, and no search of the case memory need be done, only whatever minimal search is required to match the database key. Cases are pre-indexed to permit immediate retrievals. At the other extreme, search is relied on entirely. Through an evaluation function, spreading activation, planning, blind rummaging, or some other technique, the case space is searched for the desired cases. Search may be useful even in an apparently well-indexed case base, for example, if the domain is changing rapidly, or if cases need to be retrieved in ways not anticipated or enabled by the original indexing.

The approach in this paper is to narrow the gap between an available indexing scheme and the requirements of argument through the use of best-first search guided by evaluation functions defined at various levels of abstraction. At the lowest level—the domain level—the evaluation function uses only information readily available from the indexing immediately provided by domain resource materials. At the highest level—the overall argument level—the evaluation uses information addressing the overall substance and quality of the argument. At an intermediate level—the argument piece level—the evaluation function uses information computable from the domain level but geared to the needs of the argument level.

In summary, BankXX incorporates a hybrid search-indexing approach that couples indexing with exploration of cases and other domain knowledge through best-first search in order to (1) address shortcomings in the indices

[1]This work was supported in part by the National Science Foundation, under grant IRI-890841, and the Air Force Office of Sponsored Research under contract 90-0359.

[2]An analogous vocabulary gap between instances and their generalizations has been noted by [Porter, Bareiss & Holte, 1990].

inherent in superficial domain knowledge, and (2) increase the leverage obtainable from these existing indices.

In the remainder of this section we present a capsule summary of the approach to case-base search in BankXX. The next section describes knowledge representation in the BankXX system, including the structure of the *case-domain graph.*, a graph of cases and other domain knowledge. Next, the search mechanisms used to retrieve cases are discussed. We then turn to experiments performed with each evaluation technique. A discussion of related research and a summary close the paper.

Overview of Search Model used by BankXX

BankXX creates an argument as a by-product of its best-first search of the case-domain graph informed by one of three evaluation functions. The argument is built incrementally as nodes are examined and mined for their contributions, which are collected, fit into argument components, and amalgamated into an argument. The critical aspects of the search model used by the system are summarized below and described later in detail.

Search States: Set of nodes in a case-domain graph representing either a case at some level of abstraction or a legal theory.

Initial State: (1) Problem situation or (2) user-specified node in the case-domain graph.

Operators on States: Set of functions that trace a single link or a sequence of links in the case-domain graph. Here called *neighbor methods.*

Goal States: None.

Termination Criteria: (1) Empty open list, or (2) user-specified time bounds exceeded, or (3) user-specified space bounds exceeded.

Heuristic Evaluation: Three linear evaluation functions at different levels of abstraction.

In general, state-space search is defined by a triple: *(initial state, set of operators on states, set of goal states).* In best-first search, an evaluation function is also used to guide the exploration of the state space [Barr et al., 1981]. The search performed by BankXX differs from the usual applications in two ways: the complexity of node expansions through the neighbor methods and the absence of well-defined goal states. Neighbor methods are described in the following section. We do not include goal states in our model because of the difficulties inherent in defining an "argument goal" in a way that is consistent with our informal understanding of how humans develop and evaluate legal arguments. In short, it is hard in general to say that an argument does or does not meet some plausible persuasive or rhetorical goal, or even that one has completed the supporting research.

Knowledge Representation in BankXX

Bankruptcy Domain Description

BankXX's legal domain is an area of federal bankruptcy law dealing with the issue of "good faith" in proposing Chapter 13 bankruptcy plans. Chapter 13 provides that an individual debtor with regular income may present a plan to pay off debts over time. This plan is usually the focus of a Chapter 13 proceeding, which specifies, among other things, what debts are to be paid to what creditors, and to what extent partial payment will be accepted as full satisfaction of a financial obligation. When presented with a problem situation that describes a proposed plan, the debts in issue, and other features of the bankruptcy problem, BankXX provides argument support for or against approval of the plan.

However, this approach is applicable to complex, weak-theory domains that contain a highly interconnected base of cases and other knowledge. Part of our motivation for using search to build arguments is to find a general method applicable to weak-theory domains.

Case-Domain Graph

The primary knowledge representation structure of BankXX is the case-domain graph. Its nodes represent legal cases or legal theories, and the links represent connections between them. The case base is a subgraph of the case-domain graph.

Several paths to each case in BankXX's case memory may be traversed, because cases are embedded in a graph, rather than a discrimination tree (see, e.g., [Kolodner, 1983], [Turner, 1988]). Multiple paths to cases, found through the sequential application of distinct types of indices, provide several computational advantages: they (1) can be coupled with distinct representations of cases at different levels of granularity, (2) can help resolve typical retrieval problems, such as too few or too many cases retrieved, as described in detail in [Rissland, Skalak & Friedman, 1993a], and (3) can increase the robustness of case retrieval in "real world" domains in which cases can be indexed incorrectly, since mis-indexing a case by one index does not make it inaccessible where other indices still provide a path.

Multiple Case Representations

Case nodes represent cases from each of four different points of view or levels of abstraction:

(1) **case as collection of facts** - a set of hierarchical frames capturing entry-level factual information;

(2) **case as vector of computed domain factor values** - a vector of values of factors (implemented as dimensions) computed on the facts of representation (1);

(3) **case as recurring prototypical story** - a script representing a general fact pattern or story; and

(4) **case as a bundle of legal citations** - a typed list of citations.

The implementation currently contains 186 case nodes.

Theory nodes represent legal theories as a list of factors that are necessary for determining how a particular theory applies to a case. Not yet implemented, a legal theory node also specifies a way to combine the factors — for instance, a weighting scheme — to apply the theory. Legal theory nodes are linked by pointers that describe the relationships between legal theories, such as "overlaps with," "rejects," "is derived from," and "conflicts with." (See Figure 1.) Legal theories have been culled from legal opinions by ourselves. The system contains 17 legal theories.

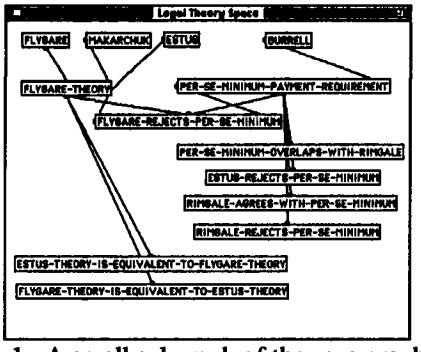

Figure 1. A small subgraph of the case graph, showing inter-theory links and links from theories to cases.

Nodes of like type are partitioned into subspaces of the case-domain graph, each of which uses a designated set of link types. For instance, in *Citation-space*, cases represented as citation-bundles are connected by citation links representing the stylized connections between a case and the cases that it cites or that cite it in various ways. The links in Citation-space are based on citation signals, such as "see," "but see," and "compare," used in legal writing ([Ashley & Rissland, 1987]), and link types, such as "affirmed," "overturned," "distinguished" used in legal databases. See [Rissland, Skalak & Friedman, 1993a] for a detailed description of the subspaces of the case-domain graph, including the legal theory space and the case spaces in which cases are represented as fact situations, domain factor vectors, prototypes, and citation-bundles.

Neighbor Methods

Neighbor methods use links in the case-domain graph to generate a list of potential nodes to examine in the search of the graph. Some neighbor methods follow pointers in a straightforward way. Others are similar to macro-operators that trace a sequence of actual links to reflect a new, virtual link[3]. For example, one simple method *(case-theory-neighbors)* gathers all the cases that have applied a theory. Neighbor methods used by BankXX include: *theory-cases-theory*, *theory-prototype*, *cited-by*, *case-theory*, *cites-case-theory*, *theory-theory*, *cites*, and *case-theory-theory-case*. These names can be interpreted as composite functions of graph nodes. For instance, the first can be thought of as the composite function *theory(case(legal-theory-node))*: find the theories applied by any of the cases that use the theory of the current node. (Cases may apply more than one theory.) BankXX has 12 neighbor methods. Some neighbor methods depend on the problem context, so that different nodes in the case-domain graph will be opened (discovered) for different problems.

Argument Data Structures

The Building Blocks of Argument: Argument Pieces.

We have chosen a simple representation of an "argument" for this implementation. In this application, an argument is a collection of *argument pieces*, which represent fragments of arguments or pieces of legal knowledge that an advocate would ideally like to have to support his or her position. The argument pieces represent building blocks of argument. We recognize that this idealization of argument does not reflect the logical and rhetorical connections between the various pieces of an argument, or the complexity of argument in general. Our immediate goal is to gather the information necessary to support a complete argument. The 12 argument pieces currently used in BankXX are:

- *cases decided for the current viewpoint*
- *best cases*[5]
- *leading cases*
- *cases sharing a large proportion of domain factors*
- *contrary cases decided for the opposing side*
- *contrary best cases for the opposing viewpoint*
- *family-resemblance-prototype*:[4]
- *supporting citations*
- *applicable legal theories*
- *nearly applicable supporting legal theories*
- *the factual prototype story category of the case*
- *factor analysis of the current problem*

Each argument piece contains a functional predicate that determines if a node can supply that useful piece of an argument. Argument pieces also contain an object slot to store entities that satisfy its predicate. BankXX builds up their content incrementally as its search proceeds, and the collection of argument pieces is output at the conclusion of BankXX's processing (Figure 2). There is no argument text generation facility within BankXX, however.

```
SUPPORTING-CASES: (<MYERS><SOTTER><MEMPHIS>
...)
SUPPORTING-BEST-CASES: (<SOTTER>)
LEADING-CASES: (<DEANS>)
DOMAIN-FACTOR-OVERLAP:
(<AKIN-FACTOR-ANALYSIS-CGN-DOMAIN-FACTOR>)
CONTRARY-CASES: (<BAEZ> <ASHTON> <CRUZ>
<OKOREEH-BAAH> <DEANS> <ALI>)
CONTRARY-BEST-CASES: (<ALI>)
SUPPORTING-CITATIONS:
(<SCHAITZ-CITES-RASMUSSEN-CGN-CITATION>
<SCHAITZ-CITES-CALDWELL-CGN-CITATION>
<SCHAITZ-CITES-RIMGALE-CGN-CITATION> ...)
APPLICABLE-SUPPORTING-THEORIES:
(<KITCHENS-KULL-THEORY-CGN-LEGAL-THEORY>
<OLD-BANKRUPTCY-ACT-GOOD-FAITH-DEFINITION>
<MEMPHIS-THEORY-CGN-LEGAL-THEORY>)
NEARLY-APPLICABLE-SUPPORTING-THEORIES: NIL
FACTUAL-PROTOTYPE-STORY: NIL
DIMENSIONAL-ANALYSIS-CURRENT-PROBLEM:
<CHURA-FACTOR-ANALYSIS>
FAMILY-RESEMBLANCE-PROTOTYPE: NIL
```

Figure 2. Example of partially instantiated argument pieces produced by BankXX for *In re Chura*, 33 B.R. 558 (Bkrtcy. 1983). "CGN" means Case-domain Graph Node.

[3]Thus the branching factor of the search space is actually greater than the branching factor of the static case-domain graph consisting of the case nodes, legal theory nodes, and link edges.

[4]The cases decided with the desired viewpoint that have the greatest family resemblance to the given case.

[5]Based on the definition of best case used in HYPO [Ashley, 1990].

Evaluating Arguments: Argument Dimensions

Just as cases may be indexed and compared on the basis of domain factors [Rissland, Valcarce & Ashley, 1984; Ashley, 1990], so arguments may be evaluated on the basis of factors that capture dimensions along which arguments may be compared and contrasted. In particular, the third type of evaluation function is based on these factors. They are also used to evaluate the "final" argument. BankXX currently uses 5 implemented *argument dimensions*:

(1) *strength-of-best-case-analogies,*
(2) *win-record-of-theory-for-factual-prototypes,*
(3) *win-record-of-theory,*
(4) *centrality-of-theory,*
(5) *centrality-of-best-cases.*

Space limitations prevent a complete description of these factors, but, for example, *strength-of-best-case-analogies* is based on the average number of legal factors common to the current problem and the best cases cited in the argument.

Indexing and Search in BankXX

In this section we discuss the operation of BankXX, and discuss in detail the three types of evaluation functions it uses to search the case base.

Control and Search Mechanisms

The control flow in BankXX is grounded in best-first search of the case-domain graph. The user specifies or the system selects a start node to begin the search. The system iterates through a loop: using the neighbor methods, compute the set of all nodes to which it is possible to move, evaluate each one of these opened nodes, select the open node with the highest evaluation function, and attempt to use that best node in some argument piece. This loop continues until a specified time limit is exceeded, a user-specified number of nodes have been opened or until the open list is empty. At the conclusion of the run, the argument pieces are collected into an argument that is assessed with respect to the argument factors. The instantiated argument pieces and the evaluation by argument dimensions are output. See Figure 3.

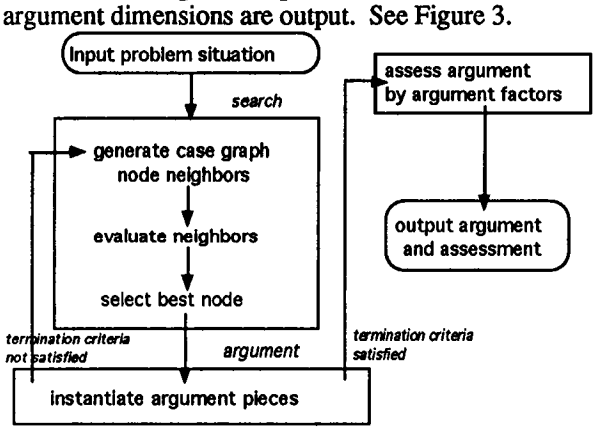

Figure 3. BankXX control flow diagram.

Best-first Search of Case Knowledge for Argument Generation

We have experimented with three types of evaluation functions that differ in the level of abstraction each uses to evaluate nodes in the case-domain graph. All the evaluation functions are simple linear functions of the form $\Sigma\ w_i\ f_i(c, *)$ where w_i are scalar weights and the f_i are scalar-valued functions of the current node c. The asterisk (*) denotes that the f_i may take additional functional arguments, described below. Since our focus is on the degree of abstraction of the terms of the evaluation function, the weights are arbitrary integers from [0, 10], except that higher weights are associated with terms involving legal theories.

The three evaluation functions applied can be analogized to those used in game-playing. By analogy, the first uses empirical game board features, such as the location of each type of piece. The second uses intermediate-level features, such as the "center-control" or "piece advantage" found in [Samuel, 1967]. The third assesses the overall quality of a completed game.

Approach 1: Evaluation at the Domain Level

In approach 1, search is guided by an evaluation function that evaluates case-domain graph nodes according to their general potential for providing information of a type known to be useful in argument. For instance, a citation-bundle node can contribute case citations, a legal theory node contributes a legal theory. This evaluation function looks only to the type of a node—cases as facts, cases as computed domain factors, cases as citation-bundles, cases as factual prototype, and legal theories—that is being evaluated.

The form of this evaluation function is:

$$w_1\ \textit{type-pred}_1(c) + w_2\ \textit{type-pred}_2(c) + \ldots + w_n\ \textit{type-pred}_n\ (c)$$

In the evaluation function of approach 1, each *type-pred*$_i$ is a function computed on features of the current case-domain graph node c. These functions are currently implemented as characteristic functions that return 1 if the domain node is of a certain class, such as a citation-bundle graph node, and 0 otherwise. Since there are five types of case-domain nodes there are five terms in this evaluation function. For the work reported in this paper, they are weighted slightly to emphasize nodes representing legal theories.

The question asked of a node with this evaluation function is: "How well will this node contribute information of a *type* known to be useful to argument?" This evaluation does not check the current state of the argument—only whether a node has the potential to contribute information known to be useful to arguments in general. It can cause the system to ignore information valuable to an evolving argument, such as the prototypical story type of the problem if one has not been found, or to pursue information less valuable overall, such as additional on-point cases when there is already a surfeit of them.

Approach 2: Evaluation at the Argument Piece Level

In approach 2, search is guided by an evaluation function that evaluates nodes according to their contributions to the system's library of argument pieces. The question asked of a node with this evaluation function is: "How well will this node contribute directly to instantiating the list of *argument pieces*?" This approach attends to progress on a wish list of argument desiderata represented by the argument pieces.

The form of this evaluation function is:

$$w_1\ arg\text{-}piece\text{-}pred_1(c,a) + w_2\ arg\text{-}piece\text{-}pred_2(c,a) + \ldots + w_n\ arg\text{-}piece\text{-}pred_n(c,a)$$

Each *arg-piece-pred*$_i$ is a two-place argument-piece predicate computed from features of the current node *c* and the current state of the argument *a*: if the *i*th argument piece can be instantiated by the current node and is *not* already filled, the predicate returns 1, and 0 otherwise. This intermediate-level evaluation function discourages BankXX from wasting computing resources through unnecessary bolstering of parts of the argument that are already established. It is more informed than approach 1 and attends to the overall evolving argument. BankXX currently uses 12 argument pieces, and thus there are 12 terms in the evaluation function.

Approach 3: Evaluation at the Argument Dimension Level

Approach 3 carries this attention to the overall argument one step further by evaluating its quality. Overall quality is assessed with the use of *argument dimensions*: factors that address desirable aspects of an argument, such as centrality of the cases cited or the coverage of cases by a legal theory. The question asked of a node by this evaluation function is: "How well will this node contribute to the *overall quality* of the argument?"

The form of this evaluation function is:

$$w_1\ arg\text{-}dim\text{-}fcn_1(c,a,a^*) + w_2\ arg\text{-}dim\text{-}fcn_2(c,a,a^*) + \ldots + w_n\ arg\text{-}dim\text{-}fcn_n(c,a,a^*)$$

Each *arg-dim-fcn*$_i$ is an argument dimension function that takes three arguments (c,a,a*), where a* is the argument that would result from incorporating the knowledge in node *c* into the current argument *a*. The argument dimension function *arg-dim-fcn*$_i$ returns 1 if the current node *c* can improve the argument along argument dimension *i*, and 0 otherwise. BankXX currently employs five argument dimensions and thus there are five terms in this evaluation function. Currently, the weights emphasize legal theories in this function as well.

The search moves to a node only if the node has the potential to make a favorable argument dimension applicable or to improve the value of an already applicable argument dimension. In approach 3, the search may be thought of as being conducted in "argument space" in that the system evaluates states representing snapshots of a partially evolved argument as constituted by the partially instantiated argument pieces and argument dimensions.

BankXX Performance Using Various Evaluation Functions

In this section, we describe the system's performance under the three evaluation functions described in the previous section: evaluation at the domain, argument piece, and argument dimension levels. Our purpose is to compare the quality of the arguments generated using the three evaluation functions, measured in terms of both the argument pieces and the argument dimensions. The data given below were collected by running each of the 54 cases in the case base as a new problem with a space resource limit of 30 visited nodes. In order to treat each case as presenting a new problem, all its links and any theories it promulgated were excised from the case-domain graph. The starting node was always the *Estus* case, the leading case in this area of the law.

Quantitative comparison of several evaluation functions

In Table I, for each of 10 argument pieces, we give the average number of nodes BankXX found by the completion of its run[6].

Argument Piece	Domain Eval. Fn.	Argument Piece Eval. Fn.	Argument Dimension Eval. Fn.
Supporting cases	5.3	3.0	9.2
Best cases	1.3	2.9	1.4
Leading cases	4.6	4.5	4.8
Overlapping factor cases	1.7	3.6	0
Contrary cases	5.4	3.0	8.3
Contrary best cases	1.2	2.1	1.4
Supporting citations	3.1	4.2	0.2
Applicable theories	1.9	1.9	0.3
Nearly applicable theories	0	0	0
Factual prototype story	0.6	0	0.2

Table I. Average number of nodes that fill 10 argument pieces.

These preliminary data suggest that the argument-piece evaluation function engenders a somewhat more balanced argument than the other two, as expected. For instance, it caused retrieved cases to be spread more evenly over a variety of types, whereas the other two over-concentrate cases in *supporting cases* and *contrary cases*. For instance, the evaluation function based on argument dimensions retrieved cases at the expense of completing other argument pieces, such as *applicable theories* and *supporting citations*.[7]

---

[6] We do not provide data for two argument pieces: the factor-analysis-of-the-current-problem piece was computed for each case, thus making its value uniformly 1, and the family-resemblance-prototype piece is being revised.

[7] It also retrieved no cases with significant domain factor overlap simply because no implemented argument dimension addresses cases that are somewhat similar to the current problem but that are not "best" cases.

Comparison of Arguments along Argument Dimensions

Arguments may be assessed for quality in a number of ways. For this experiment, we have used the argument dimensions to assess the quality of the argument ultimately generated for each case. For each of the three evaluation functions, Table II gives the average argument dimension values for the arguments created using that function. Thus the argument dimensions play two roles: they drive the search in the argument dimensions evaluation function, but are used to assess the final quality of the argument for each of the evaluation functions used.

Argument Dimension	Domain Eval. Fn.	Argument Piece Eval. Fn.	Argument Dimension Eval. Fn.
Strength of best case analogies	0.2	0.3	0.2
Theory win record for prototype	0.1	0.1	0.03
Win record of theory	0.4	0.4	0.1
Centrality of theory	4.8	4.8	0.8
Centrality of best cases	0.1	0.1	0.2

Table II. Average value along each argument dimension for the arguments created using each of the three levels of evaluation abstraction, computed over the 54 cases in the case base.

It may be surprising that the evaluation function based on argument dimensions performed somewhat less well overall than the other two functions. We speculate that the argument dimensions may be too abstract to be used alone to drive the search. Intuitively, they may "flatten" the search space and fail to grade adequately the potential utility of a node for argument. The domain and argument piece level evaluations performed comparably according to the argument dimensions. In future research we plan to extend the set of dimensions to capture more and subtler aspects of argument quality.

That no one level of evaluation stands out as best is consistent with the complexity of argument generation, which, we argue, requires that control decisions take various levels of constraints into account simultaneously [Skalak & Rissland, 1992]. While these statistics represent average results, a comparison of an argument created by BankXX with an actual judicial opinion is given in [Rissland, Skalak & Friedman, 1993b].

Related Research

Other retrieval systems have organized case memory as a graph and permit multiple paths to a case. Kolodner's CYRUS [1983] is perhaps the best classical example of a multiply-indexed case memory. Turner's MEDIC program [1988] incorporates several different types of knowledge structures in a case memory of linked discrimination nets, which allow multiple paths to diagnostic schemata. The PROTOS program [Bareiss, 1989] uses a fixed strategy for classification that takes advantage of three kinds of indexing knowledge: reminding, prototypicality and feature differences. The measure of prototypicality of an exemplar is incrementally learned by PROTOS on the basis of expert user feedback. Rose and Belew [1991] have created a hybrid symbolic and sub-symbolic system, SCALIR, that uses a variety of inter-case links that are known *a priori* (including Shepard's citation links) and applies West's key number taxonomy as citation links in a network. However, SCALIR applies a spreading numerical activation algorithm to search the network, rather than heuristic best-first search. Direct Memory Access Parsing (DMAP, [Martin, 1990]) is a case-based architecture that uses a semantic network of case frames that is searched via a marker-passing algorithm to instantiate frames that are expected in the problem context. However, none of these systems uses the constraints of argument formation — incorporated into a classical heuristic evaluation function — to drive the search of the case base.

Search has been applied in at least two systems to create arguments or to perform parts of the argument generation task [Bhatnagar, 1989] and [Branting, 1991]. Bhatnagar's algorithm involves search of a hypergraph using A* search, but depends on the determination of probability of the truth of a proposition in a potential model. Branting's GREBE uses A* search to do case matching preparatory to making an argument in the area of worker's compensation law.

The problem orientation of research into "case-based search" — how previous solution paths prune a search space [Bradtke & Lehnert, 1988], [Ruby & Kibler, 1988] — is to be contrasted with this project's research orientation. These reported experiments with case-based search used *case retrieval to guide search* for a prototypical search problem, the 8-puzzle. This project investigates the "complementary" task: how to use *search to guide case retrieval* in a general case-based problem setting.

Other systems have made use of "utility" or "cost" functions in case search and retrieval. Sycara's PERSUADER [1987] is a CBR system that created arguments in the domain of labor negotiation by searching a *belief structure*, which is a graph of the "persuadee's" goals and subgoals and their utilities. Veloso and Carbonell [1991] have studied how to balance the relative costs of search and case retrieval in a hybrid architecture combining a search-based planner with an analogical reasoner.

Summary

In this paper we have discussed how case retrieval may be performed using heuristic search of a knowledge base that uses a variety of indices and interconnections. Preliminary experiments with our BankXX system show that the paradigm of best-first search can be profitably employed in a case-based argument generation system, and that evaluation functions can help bridge the gap between simple low-level domain indices and complex higher level argument considerations. We are continuing to study the level of abstraction appropriate for the heuristic evaluation of case and domain information as well as the problem of argument evaluation.

References

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* Cambridge, MA: M.I.T. Press.

Ashley, K. D. & Aleven, V. (1991). A Computational Approach to Explaining Case-Based Concepts of Relevance in a Tutorial Context. *Proceedings, Case-Based Reasoning Workshop 1991*, 257-268. Washington, D.C. Morgan Kaufmann, San Mateo, CA.

Ashley, K. D. & Rissland, E. L. (1987). But, See, Accord: Generating Blue Book Citations in HYPO. *Proceedings of the First International Conference on AI and Law*, 67-74. Boston, MA. ACM.

Bareiss, E. R. (1989). *Exemplar-Based Knowledge Acquisition.* Boston, MA: Academic Press.

Barletta, R. & Mark, W. (1988). Explanation-Based Indexing of Cases. *Proceedings, Case-Based Reasoning Workshop 1988*, 50-60. Clearwater Beach, FL. Morgan Kaufmann.

Barr, A., Feigenbaum, E. A. & Cohen, P. (1981). *The Handbook of Artificial Intelligence.* Reading, MA: Addison-Wesley.

Bhatnagar, R. K. (1989). *Construction of Preferred Causal Hypotheses for Reasoning with Uncertain Knowledge.* Ph.D. Thesis, Univ. of Maryland, College Park, MD.

Bradtke, S. & Lehnert, W. G. (1988). Some Experiments with Case-Based Search. *Proceedings of AAAI-88, the Seventh National Conference on Artificial Intelligence*, 133-138. St. Paul, MN. Morgan Kaufmann.

Branting, L. K. (1991). Building Explanations from Rules and Structured Cases. *International Journal of Man-Machine Studies*, 34, 797-837.

Kolodner, J. L. (1983). Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(4), 243-280.

Martin, C. E. (1990). *Direct Memory Access Parsing.* Ph.D. Thesis, Yale University, New Haven, CT.

Owens, C. (1988). Domain-Independent Prototype Cases for Planning. *Proceedings, Case-Based Reasoning Workshop 1988*, 302-311. Clearwater, FL. Morgan Kaufmann, San Mateo, CA.

Porter, B. W., Bareiss, R. & Holte, R. C. (1990). Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artificial Intelligence*, 45, 229-263.

Rissland, E. L. & Skalak, D. B. (1991). CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34, 839-887.

Rissland, E. L., Skalak, D. B. & Friedman, M. T. (1993a). Case Retrieval through Multiple Indexing and Heuristic Search. *To appear, Proceedings of IJCAI-93*, Chambery, Savoie, France. International Joint Conferences on Artificial Intelligence.

Rissland, E. L., Skalak, D. B. & Friedman, M. T. (1993b). BankXX: A Program to Generate Argument through Case-Based Search. *To appear, Proceedings, Fourth International Conference on Artificial Intelligence and Law.* Amsterdam, The Netherlands.

Rissland, E. L., Valcarce, E. M. & Ashley, K. D. (1984). Explaining and Arguing with Examples. *AAAI-84, Proceedings of the National Conference on Artificial Intelligence.* Austin, TX. AAAI.

Rose, D. E. & Belew, R. K. (1991). A Connectionist and Symbolic Hybrid for Improving Legal Research. *International Journal of Man-Machine Studies*, 35, 1-33.

Ruby, D. & Kibler, D. (1988). Exploration of Case-Based Problem Solving. *Proceedings, Case-Based Reasoning Workshop 1988*, 345-356. Clearwater Beach, FL. Morgan Kaufmann, San Mateo, CA.

Samuel, A. L. (1967). Some Studies in Machine Learning using the Game of Checkers II - Recent Progress. *IBM J. Research and Development*, 11, 601-617.

Skalak, D. B. & Rissland, E. L. (1992). Arguments and Cases: An Inevitable Intertwining. *Artificial Intelligence and Law: An International Journal*, 1(1), 3-48.

Sycara, K. P. (1987). *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods.* Ph.D. Thesis, School of Information and Computer Science, Georgia Institute of Technology. Atlanta, GA.

Sycara, K. P. & Navinchandra, D. (1991). Influences: A Thematic Abstraction for Creative Use of Multiple Cases. *Proceedings, Case-Based Reasoning Workshop, 1991*, 133-144. Washington, DC. Morgan Kaufmann, San Mateo, CA.

Turner, R. (1988). Organizing and Using Schematic Knowledge for Medical Diagnosis. *Proceedings, Case-Based Reasoning Workshop 1988*, 435-446. Clearwater Beach, FL. Morgan Kaufmann, San Mateo, CA.

Veloso, M. M. & Carbonell, J. G. (1991). Variable-Precision Case Retrieval in Analogical Problem Solving. *Proceedings, Case-Based Reasoning Workshop 1991.* Washington, D.C. Morgan Kaufmann, San Mateo, CA.