High Performance Support for Case-Based Planning Applications*

James Hendler and **Kilian Stoffel**
Department of Computer Science
University of Maryland
College Park, MD.

Alice Mulvehill
Mitre Corp.
Bedford, Mass.

In this abstract we briefly describe work which uses high performance AI computing to support a transportation logistics planning application. The work particularly involves extending the retrieval mechanisms of CaPER, a case-based planning system implemented using high performance computing techniques, to support the ForMAT system being developed by researchers at Mitre Corporation.

Planning Background

In case-based planning (CBP), previously generated plans are stored as cases in memory and can be reused to solve similar planning problems in the future. CBP can save considerable time over planning from scratch (generative planning), thus offering a potential (heuristic) mechanism for handling intractable problems. With our system, CaPER, we are currently developing new approaches to CBP. In particular, one drawback of CBP systems has been the need for a highly structured memory that requires significant domain engineering and complex memory preindexing schemes to enable efficient case retrieval

In contrast, the CaPER CBP system uses high performance computing mechanisms to retrieve plans quickly from a large memory that is not preindexed. Thus it is relatively inexpensive to access memory frequently, and memory can be probed flexibly at case retrieval time. CaPER can issue a variety of queries that result in the retrieval of one or more plans (or parts of plans) that can be combined to solved the target planning problem. These plans can be merged and harmful interactions among them resolved using annotations on a plan that capture interdependencies among its actions (8; 7; 5; 9).

*This research was supported in part by grants from NSF(IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), the ARPA I3 Initiative (N00014-94-10907) and ARPA contract DAST-95-C0037. Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384).

Transportation Logistics Planning

The United States Transportation Command (USTRANSCOM) is responsible for generating and maintaining the plans by which United States military forces are deployed. This responsibility includes determining the transportation needs for missions short and long, small and very large. For large missions, the process by which these transportation plans are constructed can be very complex and time consuming. Representatives from the various services and commands involved in a plan must collectively decide how best to allocate the limited transportation resources (aircraft, ships, trucks and trains) to achieve the many military goals of the mission. The end result of this process is an Operational Plan (OPLAN) which specifies where and when the forces involved in a mission are to be moved. Associated with a OPLAN are one or more Time Phased Force Deployment Data (TPFDD) which describe what, when, and how the forces for a mission will be deployed. The OPLAN and TPFDDs are stored and maintained until their execution is called for. At this time the plan will generally have to be modified to fit the particular details of the current situation.

ForMAT (Force Management and Analysis Tool) provides an environment in which a force development and planning specialist can view, modify, and create the basic structures of TPFDDs (called force modules, or FMs). An FM prescribes a force or set of forces that can be used to satisfy some planning requirement. The FM is typically a grouping of combat, combat support, and combat service support forces, and ranges in size from the smallest combat element to the largest combat element. It may specify accompanying supplies and the required movements, resupply, and personnel necessary to sustain forces for a minimum of 30 days. The elements of a FM are linked together so that they may be extracted from or adjusted as an entity to enhance the flexibility and usefulness of a plan. One or more FMs for use in a given plan are stored in a TPFDD. In theory, FMs form a library which can be drawn upon to quickly build a new plan. In a crisis, new TPFDDs will be built, at least in part, from FMs

within one or more existing TPFDDs.

The force modules that compose TPFDDs are themselves composed of smaller units called Unit Line Numbers (ULNs). A ULN identifies a force, support for a force, or a portion of a force. A ULN is often described by its Unit Type Code, which can span a wide range of items from tactical fighter squadrons and army battalions to dog teams, or even a Catholic chaplain. Finding appropriate ULNs (and therefore FMs) in previous TPFDDs is a complex task, similar to case-retrieval in case-based planning. (For more detail of the ForMAT system, see the paper by Alice Mulvehill in this volume.)

High Performance support for TPFDD retrieval.

Joint work between MITRE and UMCP is focusing on the potential to use the case-retrieval subsystem of CaPER as part of the ForMAT system. Working either interactively, with a human planner, or as an agent within a larger mixed-initiative planning system, we hypothesize that the CaPER retrieval methods will scale well to exploring hybrid knowledge-base systems which can scale well to the requirements of retrieval even within very large information systems containing information about a large number of TPFDDs.

We are currently working on providing a close connection between ForMAT and CaPER, and trying to assess the benefits of the parallel query mechanisms. Current experiments focus on using parallel processors in support of a large knowledge base containing information about 17 TPFDDs, representing a total of 322 Force Modules and over 14,000 unit line numbers.

To put the size of this into perspective, we previously encoded one large TPFDD into the knowledge representation system: MITRE provided a database of the FMs and ULNs, and at UMCP a program was written to recode it in the frame-based system underlying CaPER. This TPFDD had about 6,000 ULNs. The TPFDD required 6,300 frames and expressed approximately 87,000 assertions among them. In addition, a domain-specific ontology was created containing approximately 1,000 frames for domain concepts such as "FM", "ULN", service branch, military units, weapon types, capabilities, etc. Frames in the base ontology were organized in an abstraction ("is-a") hierarchy. The new TPFDD KB, with the full 17 TPFDDs is expected to be 4-5 times the size of the previous one, with a richer ontology as well.

Previous Technology Interaction Experiment

To test feasibility, and to assess the speed and scalability of the parallel mechanisms, a number of random queries were formulated and retrieval was performed using a fast serial algorithm running on a Macintosh Quadra 850 with 24M of Ram and the parallel algorithm running on the parallel CM-5 with 32 processors. The queries included recognition queries of the form "find all x with properties p1, p2, ... , pn" and structure queries of the form "find all x and y... with relations r1(x,y), r2(x,y), ..., rn(x,y)..." Recognition queries have a single variable while structure queries have multiple variables. For example, recognition queries included "Retrieve all FMs that belong to the Air Force, have a maintenance function, have combat air capability, and have a POD-source of 'in-place'" and "Retrieve all FMs attached to the Army that have a maintenance function and are destined for Sri Lanka." Structure queries included "Find 2 ULNs from the same service with the same destination" and "Find 2 ULNs from different services with the same destination and point-of-departure modes."

For the recognition queries, serial retrieval time varied from about 1 second to as much as 100 seconds. Parallel retrieval time remained below 1 second. For the structure matching queries, serial time ranged from about 90 seconds to as much as 1200 seconds (about 20 minutes). Parallel time remained under 5 seconds for all queries. These early results clearly indicated that the parallel retrieval algorithms performed well, and scales well to larger problems much better than serial algorithms. Based on these experiments, the decision was reached to expand the capabilities of the system, and to work towards a closer joining of ForMAT and CAPER.

Research Emphasis - High Performance Support for ForMAT

The testing of the parallel system revealed a number of places where performance improvements could occur. In addition, although the times of the initial experiments were encouraging, ongoing work on the knowledge representation system convinced us we could do substantially better. In particular, CAPER runs on top of a frame-based system called Parka. We have now reimplemented Parka to be able to run on a wide variety of generic supercomputers, as well as on smaller workstations and/or networks of personal computers (using an MPI protocol). The new version, running on an IBM SP2 with 16 processors, is approximately two orders of magnitude faster than the CM5 implementation. We have currently reimplemented all the CAPER functionality, especially the structure matching components, and are now reintegrating the system with the format system (see section). In the remainder of this section, we first describe experiments aimed at showing that the new version of Parka is substantially faster and more scalable than the old, thus making it more able to support the ForMAT system. In the second part, we discuss structure matching queries (the main queries needed to properly support ForMAT) and how they scale in the new implementation.

Testing recognition queries We have performed a number of tests on large KBs encoded in the PARKA system. The largest one we had used previously in testing the SIMD version of the system was the ontology of the CYC knowledge base developed at MCC (6). As previously reported in (4), we encoded the ontology of version 8 of CYC into Parka. In our version, this ontology has about 32,000 frames and about 150,000 assertions relating them (property and ISA links). The biggest difference between the generated networks and CYC is that the test network had only a single property on the root, thus making all "subgraphs" equal to the entire KB. In CYC the ISA hierarchies under particular properties are only subnetworks of the full 32,000 frames.

We hoped to directly compare the SIMD and MIMD versions using queries on the CYC system, but most of the tests we'd run previously were, basically, too easy for the new version. In particular, the subnetworks in CYC are all much smaller than the smallest test network described in the previous section. For this reason simple queries like "What's the color of frames X" could not be used to show parallel effects – their time was on the order of 50-100μseconds on a single processor! Thus, instead of timing single inheritance queries, we generated a new class of recognition-like queries that were designed to stress the new system.

To start with, we tried the timing of recognition queries in some of CYC's largest subtrees. The results were quite promising, but again to fast for showing parallel speedups. For example, using the scan algorithms, we executed some recognitions with more then twenty conjuncts and had single processor response times of under 1/100 of a second (compared with about 1 second for the SIMD system).

To further stress the scanning algorithms and to explore speedups, we designed a new set of inferences specifically to that purpose. In particular, we used queries that would include great amounts of both inheritance and recognition. These queries were of the following form: "Give me all the frames which have one or more properties in common with frame X". To be sure to get "slow" times, we ran these queries on frames in big ISA sub-hierarchies of CYC. Two of the biggest, plant and animal, were chosen for testing, since they had very large numbers (in the tens of thousands) of other frames which shared at least one property.

The execution time for the queries "Give me all the frames with the same properties as Animal" and " Give me all the frames with the same property as Plant" are presented in figure , which compares these times to the optimal speedup curve.

As one can see the recognition algorithm behaves very well. The efficiency is about 75%. That's very high in respect to the small amount of CPU time needed for the inheritance algorithm. The queries represented in the figure are much more complex then standard queries in CYC.

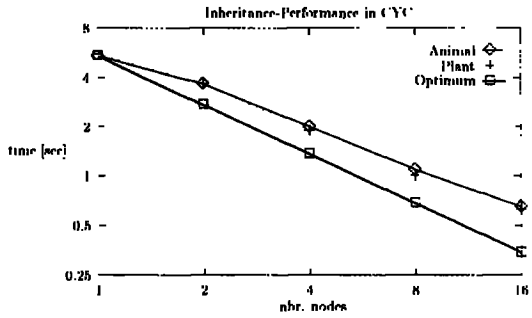

Figure 1: Recognition Performance in CYC

Representing structures for matching

Our description of the problem of structure matching follows that given in (10). A knowledge base defines a set, P, of unary and binary predicates. Unary predicates have the form $P_i(x)$ and binary predicates have the form $P_j(x_1, x_2)$, where each x_i is a variable on the set of frames in the KB. An existential conjunctive expression on these predicates is a formula of the form $\exists x_1, \ldots, x_m : P_1 \wedge P_2 \wedge, \ldots, \wedge P_n$, where $n \geq 1$. Our task is to retrieve all structures from memory which match a given conjunctive expression. Therefore, we would like to find all such satisfying assignments for the x_i.

We can view the problem of matching knowledge structures in two ways. The first is as a subgraph isomorphism problem [1]. We view variables as nodes and binary predicates as edges in a graph. We want to find structures in memory which "line up" with the graph structure of the query expression. The other way to view the matching problem is as a problem of unification or constraint satisfaction. If we can find a structure in memory which provides a consistent assignment to the variables x_i (i.e., unification), then that structure matches the conjunctive expression.

Overview of the algorithm The structure matching algorithm operates by comparing a retrieval probe, P, against a knowledge base (KB) to find all structures in the KB which are consistent with P. This match process occurs in parallel across the entire knowledge base. A Parka KB consists of a set of frames and a set of relations (defined by predicates) on those frames. Most relations are only implicitly specified and so must be made explicit by expanding the relation with the appropriate inference method. By computing inherited values for a relation, all pairs defining the relation are made explicit. We currently allow only unary and binary relations.

A retrieval probe is specified as a graph consisting of a set of variables $V(P)$ and a set of predicates (or constraints) $C(P)$ that must simultaneously hold on

[1] More specifically, this is a problem of sub-DAG isomorphism with typed edges, the edges being the relations in the KB between frames.

frames bound to those variables. The result of the algorithm is a set of k-tuples, where each k-tuple encodes a unique $1-1$ mapping of frames to variables in $V(P)$, that unifies with the description of the structure in memory with $C(P)$. Figure 1 shows a simple example of the structure matching algorithm; given a knowledge base represented as a semantic network and a probe graph, the algorithm finds all matching subgraphs in the network which are isomorphic to the probe.

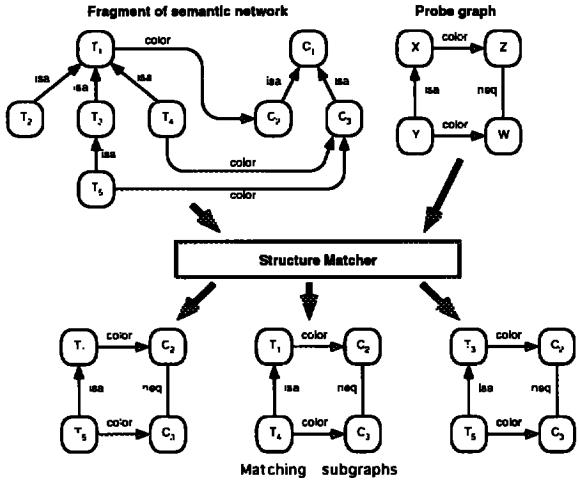

Figure 1: A simple example of structure matching.

The set of frames that can bind to each variable is initially restricted by a set of constraints indicated by unary predicates. Each unary constraint may only constrain the values of one variable. Examples of these constraints are "X is a dog" or "the color of X is yellow". We allow set theoretic combinations of the unary constraints, for example "X is a dog and the color of X is yellow", or "X is a dog but X is not yellow"[2] The domains for each variable are maintained throughout the match process and are further restricted as more constraints are processed.

Constraints between frames bound to variables are specified by a set of binary constraints. For example, we can say "the color of X must be Y", or "X is a part of Y", for some X and Y in $V(P)$. Binary constraints are processed by "expanding" the binary relation given in the constraint. By expansion we mean that all pairs participating in a relation R in the KB are made explicit by invoking the inference method for the associated predicate. The pairs allowed to participate in the expanded relation are restricted to those in the domains of the variables related by R. For example, a binary constraint may be expressed as (Color X Y). In this case the values for each concept in the domain of X are computed for the color predicate and pairs that have values outside the domain of Y are excluded. Two additional binary predicates, "eq" and "neq" are provided to provide codesignation and non-codesignation of variables. These constraints act as a filter, eliminating any tuples from the result for which the constrained variables are(not) bound to the same frame.

The result of a structure match is a set of k-tuples, where each tuple corresponds to a satisfying assignment of the k variables. Alternatively, the result can be viewed as a relation. Initially, the matcher begins with an empty set of relations. During the match, several intermediate relations may be constructed. Simple binary relations result from the expansion of a binary constraint. These are later fused (via a relational join operation) or filtered (via codesignation or non-codesignation) until a single relation remains. The algorithm selects binary constraints to process using a greedy algorithm based on a simple cost model stored in the metadata.

Testing Structure Matching Queries A second domain used for testing the MIMD implementation was on knowledge bases that were created as part of doctoral thesis work on the CAPER system itself (8; 7). Part of this project involved the automatic seeding of large case memories by a generative planner. One domain used in this work was the "UM Translog" domain, a logistics planning domain developed for the evaluation and comparison of AI planning systems. Case-bases of various size were created, and each contains a number of plans, derivation information, and planning related ontologies.[3] To measure the performance of the strucure matcher we used the UM-Translog KB in different sizes (10 cases, 100 cases and 200 cases).

The results presented in this section are especially interesting for two reasons. On one hand we are presenting the timings for the structure matcher, the most complex retreival algorithm implemented in PARKA to date. On the other hand we are also able to show, that the new PARKA system has the capability to handle verly large KBs on a single Sparc workstation.

As the planning system solves a problem, we store all the queries that are generated. For testing the parallel system we chose several queries at random from a large number of stored queries. The results are summarized in Table 1. Table 1a presents the timings for the six queries on a SPARC 20 using the three different KBs, and Table 1b presents the timings on 1, 8 and 16 nodes

[2] Variables in the query probe which do not appear in a unary constraint are treated differently. Variables not contained in a unary constraint are still able to be constrained by intersecting the instances of the range and domain of the predicates in binary constraints in which the variable appears.

[3] A full description of the domain, the complete specification of the planning operators used, and the case-bases themselves are available on the World-Wide Web at *http://www.cs.umd.edu/projects/plus/UMT/*.

Query	20 CB	100 CB	200 CB
1	1020	4740	6990
2	195	1305	1635
3	225	1470	1725
4	630	3570	4590
5	675	3600	4605
6	405	585	645

a: Timings (milliseconds) on a SPARC 20 for a 20, 100 and 200 cases KB

Query	1	8	16	1:8	1:16
1	3041	546	313	69.6	60.7
2	713	129	75	69.1	59.4
3	753	135	79	70.0	56.6
4	1997	361	205	69.1	60.9
5	2003	360	206	69.5	60.8
6	284	52	29	68.2	61.2

b: Timings (milliseconds) on a 1, 8 and 16 nodes of an SP2 on 200 case KB

Table 1: UM-Translog Timings (serial and parallel).

cases	frames	structural links	property links
20	11612	26412	114359
100	59915	130558	800481
200	123173	266176	1620456

Table 2: Sizes of the UM-Translog KBs.

of an SP2 using the 200 case KB (the largest of the three). The actual sizes of these KBs are shown in Table 2, where frames is the number of nodes in the DAG, structural links are those in the ISA ontology, and property links are all others (i.e. the number of the edges in the DAG is equal to the structural links plus the property links). (We believe the 200 case case-base to be the largest meaningful semantic net created to date, as can be seen it contains about 1.8M assertions concerning over 123,000 entities.)

As can be seen, the sequential timings range from under a second for the simplest query to about 7 seconds for the most complex. On the parallel system all queries were executed in under one second, with the simplest query taking only 29 milliseconds on 16 processors, and the most complex taking only 303 millisecond. Table 1b also shows the efficiencies of the parallel algorithm. Even in the current non-optimized form, the efficiency averages about 69.3% for eight processors and 59.9% for 16 processors.

Current Directions

Curreently, the goal of this joint project is focusing on providing a closer integration between the UMCP high performance case-based system and the MITRE ForMAT tool. Current directions include:

- The parallel case-based retrieval system is implemented on top of a frame-based system called Parka, which has a complex syntax for expressing recognition and structure queries. We are working to develop a C-based API which will allow functions to be invoked either locally (on a Sparc) or remotely (on a parallel machine), This will provide a mechanism allowing ForMAT and other applications to directly run queries using the Parka High Performance Support.
- The set of queries used to test the system was, as stated previously, randomly generated. Current work focuses on identifying specific queries and query classes that take too long to service on the serial MITRE system. The development of parallel algorithms for these query classes, and the testing thereof, will be a focus for research in the coming year. In particular, queries used in to test Format in military planning exercises will be recorded and used for this research.
- To facilitate a closer integration between ForMAT and the retrieval system. a second API is being defined to allow a Parka browsing tool to run on knowledge bases defined using other KR languages (including the internal language used in ForMAT). This will allow us to more precisely define the relations between Parka and ForMAT.

References

Andersen, W., Evett, M., Hendler, J. and Kettler, B. "Massively Parallel Matching of Knowledge Structures," in *Massively Parallel Artificial Intelligence*, Kitano, H. and Hendler, J. (eds.), AAAI/MIT Press, 1994.

Evett, M.P., Hendler, J.A., and Spector, L., "Parallel Knowledge Representation on the Connection Machine," *Journal of Parallel and Distributed Computing*, 1994.

M.P. Evett. *PARKA: A System for Massively Parallel Knowledge Representation*, Ph.D. thesis, Dept. of Computer Science, University of Maryland, College Park, 1994.

Evett, M.P., Hendler, J.A., and Andersen, W.A., "Massively Parallel Support for Computationally Effective Recognition Queries", *Proc. Eleventh National Conference on Artificial Intelligence*, 1993.

Hendler, J. High Performance Artificial Intelligence, *Science*, Vol. 265, August, 1994.

Lenat, D.B. and Guha, R.V., "Building Large Knowledge-Based Systems", Addison Wesley, Reading, Mass., 1990.

Kettler, B.P., Hendler, J.A., Andersen, W.A., Evett, M.P., "Massively Parallel Support for Case-based Planning", *IEEE Expert*, Feb, 1994.

Kettler, Brian "Case-based Planning with a High-Performance Parallel Memory," Doctoral Dissertation, Department of Computer Science, University of Maryland, October, 1995.

K. Stoffel, J. Hendler and J. Saltz, High Performance Support for Very Large Knowledge Bases, *Proc. Frontiers of Massively Parallel Computing*, Feb, 1995 (Extended Abstract).

Watanabe, L., and Rendell, L., "Effective Generalization of Relational Descriptions", AAAI Eighth National Conference on Artificial Intelligence. 1990.