Enabling Cognitive Architectures for UAV Mission Planning

Jon C. Russo, Mohammed Amduka, and Boris Gelfand, Lockheed Martin Advanced Technology Laboratories
{jrusso, mamduka, bgelfand}@atl.lmco.com

Dr. Keith Pedersen, Lockheed Martin Aeronautics
keith.pedersen@lmco.com

Dr. Richard Lethin and Dr. Jonathan Springer, Reservoir Laboratories
{lethin, springer}@reservoir.com

Dr. Rajit Manohar, Cornell University
rajit@cs.cornell.edu

Dr. Rami Melhem, University of Pittsburgh
melhem@cs.pitt.edu

Abstract

The operational performance desired for autonomous vehicles in the battlefield requires new approaches in algorithm design and computation. Our design, Polymorphic Cognitive Agent Architecture (PCAA), is a hardware-software system that supports the requirements for implementing a dynamic multi-unmanned aerial vehicle (UAV) mission planning application using cognitive architectures. We describe the requirements for our application, and discuss the challenges of using current "non-cognitive" algorithms to solve this problem and the reasons this motivates our experiment. We describe our system to integrate multiple cognitive architectures. We identify the cognitive kernels used in the cognitive and non-cognitive implementation and system software and hardware features to support them. We also discuss observations and plans for further development and evaluation.

Dynamic Multi-UAV Mission Planning

The problem of coordinating multiple UAVs has been considered by several authors [9]. Balancing targets and threats, capabilities and constraints, and policy/tactical issues must be factored into the planning problem. For example, planning must account for constraints such as separating ingress and egress points, maximizing covertness (by minimizing radar cross-section during transit), and avoiding flying long straight-and-level route segments. Fielded systems for manned aircraft static planning work roughly as follows, in an algorithmic manner. Implementations typically represent as many of these considerations as possible as costs on the edges of a graph, where the nodes in the graph correspond to particular discrete choices of position, orientation, or controls. The presence or absence of an edge indicates the physical feasibility of transition between the nodes. Considerations that cannot be framed as costs overlay the graph, so that the planning problem is essentially constrained shortest path finding. Good polynomial time algorithms exist for shortest path finding. The facts that the graphs are large, have high dimensionality, and the computation of edge costs is compute intensive (e.g., for radar cross section) are by themselves sufficient to make this problem challenging, but the presence of overlaying constraints makes the problem NP-hard. Consequently, to field systems, developers simplify the problem by phasing the optimization through different subsets of costs and constraints that progressively refine approximate plans and by paying careful attention to graph construction (e.g., excluding regions that are unreachable due to fuel constraints).

When this solution is extended to the multi-UAV domain, the requirements for computation grow exponentially. The major contributor to this growth is that the dimensionality of the planning space is proportional to the number of UAVs; thus, the size of the graph to be searched grows exponentially with the number of UAVs. This is further multiplied by the concurrent desire to plan over larger physical regions, at finer granularities, and with more complicated constraints arising from considerations such as route deconfliction and more complex tactics.

Furthermore, imposing the requirement that the system be able to re-plan dynamically (for example, in the face of pop-up threats or in-flight re-tasking) implies that such planning be completed in seconds. Our estimates are that the power dissipation to compute plans in such time using current problem framing techniques and conventional processor technology is in the range of tens of Megawatts. Such an approach is impractical, either for on-board processing or for external processing in ground stations.

Cognitive Approach

We are attempting to apply techniques arising from the "cognitive architecture" research program [4] to the problem of UAV path finding. Cognitive architectures draw from our increasing body of experimental observations of human cognition and theories about intelligence. A cognitive architecture unifies these insights and theories and renders them in computer based models. These models can exhibit aspects of intelligent behavior such as

This work was sponsored by DARPA/IPTO in the Architectures for Cognitive Information Processing program; contract number FA8750-04-C-0266. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

reasoning, learning, and decision making. One objective of research into cognitive architectures is that such intelligence could exhibit generalized problem solving ability. While the quest for generalized problem solving ability is an early facet in the history of artificial intelligence [6], the accumulated body of results in this program is attracting a number of researchers facing the need for complex automated problem solving in diverse fields (e.g., [2]).

Cognitive Framework

We have chosen to experiment with the SOAR [7], ACT-R [1], and swarming [8] models of cognition. SOAR is based on production systems in service of explicit goals; in each execution cycle, all productions whose precondition is enabled by the content of memory will fire, potentially updating the working memory or rule base. SOAR has designed-in mechanisms for resolving conflicts and integrating knowledge from multiple sources. ACT-R emphasizes reactive problem solving through expertise-based pattern matching. In our system, ACT-R also provides the interface between the macro-cognitive symbolic operations of SOAR and the proto-cognitive sub-symbolic operations provided by swarming computation. Using hierarchical ant clustering, swarming implements perception processing, extracting cognitive elements from massive amounts data.

Discussion of Cognitive Approach

There is an apparent correspondence between our chosen cognitive frameworks and the UAV mission planning problem as framed algorithmically that arguably should provide problem simplification. For example, swarming is a well-known meta-heuristic used in the solution of graph problems. The chunking operation in SOAR is an Explanation-Based Learning procedure that is analogous to the conflict clause generation procedure found in fast constraint solvers. The caching and approximate matching in ACT-R can leverage previous knowledge of good solutions that simplify the generation of new plans. But we are more ambitious in our desire to apply the cognitive frameworks; for example, we would like to investigate having SOAR reasoning about the problem in terms of an upper ontology for high level problem simplification.

Our architecture is also an experiment into the fusion of the different cognitive models into one intelligence; we have prototyped a cognitive markup language (mCML) to implement communication between the intelligences. Finding ways to use the heuristics power of swarming or ACT-R to make the reasoning in SOAR tractable is just one of the numerous opportunities for further investigation of such a configuration. Furthermore, we would like to investigate integrating other types of cognitive models into the fusion, such as for probabilistic reasoning to model uncertainty or subsumption architectures for greater integration of the UAV control system.

Another challenge is the degree and manner of integration of the existing algorithmic approaches to mission planning with the cognitive architectures. It is an open question which is better: a fine-grained fusion of such techniques (implementing the search directly in one of our frameworks) versus coarse partitioning (coupling a highly tuned constrained A* solver with a highly tuned implementation of a cognitive architecture). An implementation of A* in our cognitive frameworks, for example, is attractive from the point of view of a seamless transition between that path finding and the high level reasoning, but it could be computationally very inefficient.

Computing Requirements

We have identified the following cognitive kernels as being components of the UAV mission planning problem: minimum-cost path planning, constraint satisfaction, assignment problems, clustering, TSP solver, knowledge-based reasoning, and probabilistic reasoning. Mindful of the fact that examining these kernels in isolation gives only partial insight into the types of system and hardware architectures needed to facilitate such processing in a complete application implementation, we describe below three of several approaches we have investigated for such kernels.

Hardware Support for ACT-R

ACT-R employs memory intensive operation for expertise-based solution using three logical memory compartments: the declarative memory is used to symbolically represent compositional data, the procedural memory stores information for reaction in dynamic problem solving, and the working memory maintains the current operational context. Each memory type operates on similar principles for matching. The match operation is inexact in that the data associated with the closest key in memory (according to a definable distance metric) is retrieved. The similarity function, which compares an input operand to each content row (or chunk) of memory may be further compounded by factors such as decay or practice.

Our hardware designs for ACT-R leveraged the inherent parallelism in the independent comparisons of chunks with input operands. Figure 1 shows the results of speedup versus number of Power PCs cooperating on the match operation.

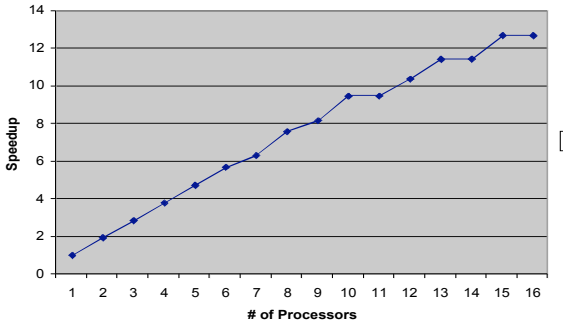

Figure 1: Speedup of ACT-R versus number of processors

Further speedup (up to two orders of magnitude) was achieved by using a custom inexact matching architecture (implemented in a Xilinx Virtex-2 FPGA) shown in Figure 2.

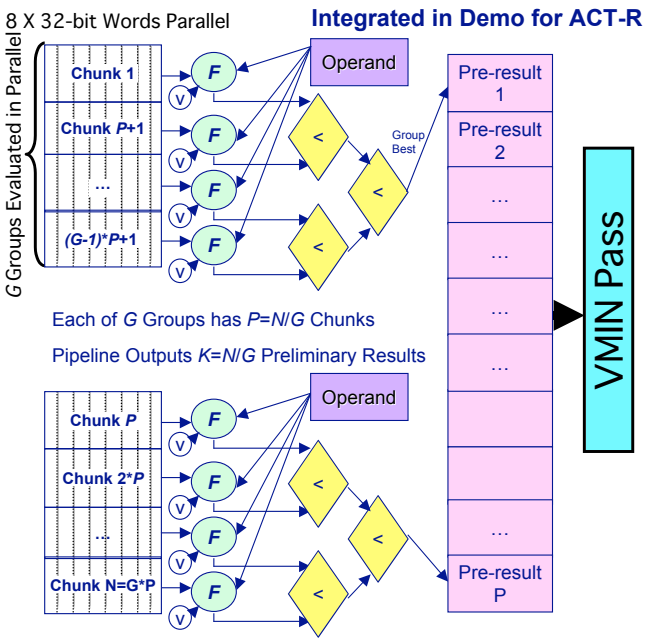

Figure 2: Matching memory for ACT-R acceleration

The programmable match memory of Figure 2 implements a match function (F) in parallel across 256 bits of the input operand, along with parallel banking of memory, objective evaluation, and result selection. The implementation can sustain in excess of a 50 mega-chunks-per-second match rate for a fully-pipelined match function.

Hardware Support for SOAR

The core production matching of SOAR targeted for hardware implementation is based on an algorithm known as Rete [3]. Rete is an exact-match algorithm that uses two memory types: the *working memory* contains facts about the world that are collected over time, and the *production memory* contains rules. Each rule in the production memory would typically be a set of conditions and a set of actions to perform if those conditions are met. An added complexity is that the condition expressions may contain variables that must be bound during matching.

The study of Rete hardware targets a simulation of the event driven processing afforded by asynchronous logic [5]. Figure 3 shows an asynchronous data flow network for a Rete, where each node is built from fine-grained logic and can execute concurrently as soon as dependencies are met. As is typical with asynchronous approaches, energy consumption for computation is reduced by computing only when necessary, and performance can be greater by removing the need for a central clock.

Due to the dependencies of typical networks, the maximum concurrency achievable is only on the order of 10X to 20X. Hardware support for hash functions yields higher constant factors of acceleration and are under consideration.

System Support for Swarming

To support swarming, a key cognitive component of the PCAA architecture, we attempted to abstract the common control aspects of swarming algorithms through development of an API. Whereas swarming algorithms were traditionally written as custom applications, often from scratch, this API is useful to allow more sophisticated use of the underlying hardware. By handling low-level details, this reusable infrastructure also naturally helps support rapid cognitive application development.

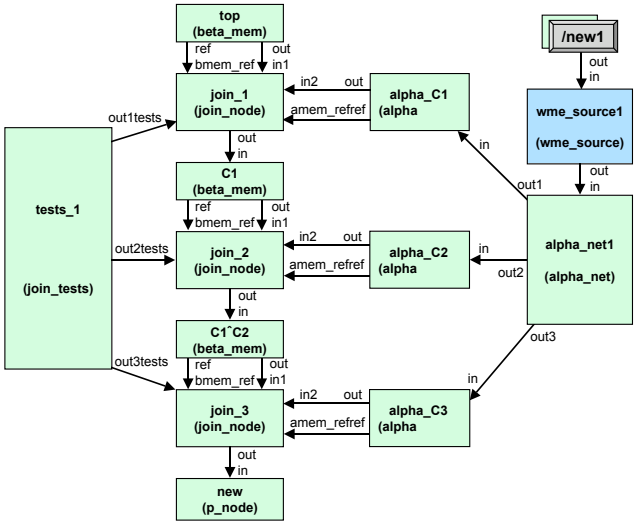

Figure 3: Rete in event-driven logic

Swarming is conceptually highly parallel. In our API prototyping, we found this parallelism to be readily available in practice. The challenge is often the grain size; however, with too little work per node, communication costs can dominate. The grain size issue through clustering of highly-connected nodes and communication-aware layout of work, including dynamic rebalancing, will be important to manage.

Further Discussion

The results of these studies show that specialized hardware can significantly outperform general purpose processing (in some cases by orders of magnitude) for kernels of our chosen cognitive framework. However, along with Amdahl's law, there are additional obstacles to achieving system level performance on par with the speedups of the individual critical components. For example, the transformation of sensory data into a symmetry-invariant form suitable for lookup in a limited memory can be resource intensive and difficult to generalize.

To help system designers select cognitive processors, metrics are being developed. An indicator of performance identified here is "chunks-per-second," but in general, measures of performance of a cognitive system may not be so clear cut and may not apply to different frameworks.

The architecture of cognition is a continuing research program, and the attempt to apply the general methods to this specific application should further extend our understanding of these frameworks.

The continued top-to-bottom implementation of this application using cognitive techniques should yield additional insights and capabilities. For example, while we have begun to study the kernels and cognitive frameworks

in isolation, the integrated application should provide insights and opportunities for specialized operators working within application-compatible reduced reliability or exploiting temporal or spatial locality in the control and data flow across kernels. The cognitive frameworks are, in a sense, interpreters running a high-level cognitive program; with more insight into the structure and behavior of this program, we expect to be able to develop specialized system and hardware mechanisms to exploit them. System and hardware support that reduces the cost of using cognitive frameworks should facilitate further application of cognitive techniques, increasing the intelligence of important embedded systems.

Acknowledgement

The authors would like to thank DARPA, AFRL and government laboratories for fostering innovation in this area.

References

[1] J. R. Anderson and C. Labierre. *The Atomic Components of Thought,* Lawrence Erhlbaum 1998.

[2] D. D. Clark, C. Partridge, J. C. Ramming, J. T. Wroclawski, "A Knowledge Plane for the Internet", in Proceedings of the 2003 Conference on Applications Technologies, Architectures, and Protocols for Computer Communications, 2003.

[3] R. B. Doorenbos, *Production Matching for Large Learning Systems,* Ph. D. Thesis, CMU, 1995.[

4] P. Langley, J. E. Laird, S. Rogers, "Cognitive Architectures: Research Issues and Challenges," Unpublished, cited by permission, 2006.

[5] R. Manohar and K. Mani Chandy. "Delta-Dataflow Networks for Event Stream Processing," Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems, November 2004.

[6] A. Newell, J. C. Shaw, H. Simon, "Report on a General Problem-Solving Program," In Proceedings of the International Conference on Information Processing, 1960.

[7] A. Newell, *Unified Theories of Cognition,* Harvard University Press, 2000.

[8] H. V. D. Parunak, S. A. Brueckner, and J. Sauter, "Digital Pheromones for Coordination of Unmanned Vehicles," In Proceedings of Workshop on Environments for Multi-Agent Systems (E4MAS 2004), Springer, 2004.

[9] A. F. Wehowsky, *Safe Distributed Coordination of Heterogeneous Robots through Dynamic Simple Temporal Networks*, Ph.D. Thesis, MIT, 2003.