Published in Leake, D.B., editor (1996), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, AAAI Press.

MULTI-PLAN RETRIEVAL AND ADAPTATION IN AN EXPERIENCE-BASED AGENT

Ashwin Ram and Anthony G. Francis, Jr.
College of Computing, Georgia Tech

1. Introduction

The real world has many properties that present challenges for the design of intelligent agents: it is dynamic, unpredictable, and independent, poses poorly structured problems, and places bounds on the resources available to agents. Agents that operate in real worlds need a wide range of capabilities to deal with them: memory, situation analysis, situativity, resource-bounded cognition, and opportunism. In particular, agents need the ability to dynamically combine past experiences to cope with new situations, selecting and merging the relevant parts of remindings to take maximum advantage of their past experience.

To address these issues, we propose a theory of *experience-based agency* which specifies how an agent with the ability to richly represent and store its experiences could remember those experiences with a context-sensitive, asynchronous memory, incorporate the relevant portions of those experiences into its reasoning on demand with integration mechanisms, and direct memory and reasoning through the use of a utility-based control mechanism. We have implemented this theory in the NICOLE multistrategy reasoning system and are currently using it to explore the problem of merging multiple past planning experiences. NICOLE'S control system allows memory and reasoning to proceed in parallel; the asynchronous and context-sensitive nature of that memory system allows NICOLE to return a "best guess" retrieval immediately and then to update that retrieval whenever new cues become available.

But solving the problem of merging planning experiences requires more than just memory and control. We need mechanisms to integrate new retrieved plans into the planner's current reasoning context whenever they are found; to ensure that those new retrieved plans are useful, we need ways to use the planner's current context to generate new cues that can help guide the memory system's search. To solve these subproblems, we have developed the Multi-Plan Adaptor (MPA) algorithm, a novel method for merging partial-order plans in the context of case-based least-commitment planning. MPA allows the merging of arbitrary numbers of plans at any point during the adaptation process; it achieves this by dynamically extracting relevant case subparts and splicing those subparts into partially completed plans. MPA can also help guide retrieval by extracting intermediate goal statements from partial plans.

In this chapter, we briefly review the properties of the real world that present challenges for the design of intelligent agents, examining in particular the need to combine past planning experiences. We then review our theory of experience-based agency and its implementation in the NICOLE system. We present the MPA algorithm, illustrate how it supports the merging of plans at any point during the adaptation process, and describe its foundations in least-commitment case-based planning. We then discuss how MPA can be integrated into various

control regimes, including systematic, pure case-based, and interleaved regimes, and describe our implementation of interleaved MPA in NICOLE. We conclude the paper by reviewing other case-based planning work and then outlining our contributions.

2. Exploiting the Past

2.1. The Challenges of the Real World

The real world presents many challenges to an agent, challenges which arise in both artificial domains and in the real-world problems humans face (Bratman, 1987; Hammond, 1989; Orasanu & Connolly, 1993; Pollock, 1995; Sternberg, 1985, 1986; Wooldridge & Jennings, 1995). The real world is *dynamic*, changing independently from the actions of an agent, and it is *unpredictable*, changing in a way too complex for an agent to completely predict. To make things worse, the world and its changes are *relevant* to the agent's goals (so that they cannot be ignored), *sensitive* to the agent's actions (so that the agent cannot act with impunity) and place *resource bounds* on the agent's activity (so that the agent cannot just try everything until it works).

But the real world is also *regular*: patterns exist, encapsulating classes of objects and events and relationships of cause and effect. An agent's ability to effectively use its past experience with these patterns can be key to its successful performance and survival in real-world domains. In this chapter, we take as a given the traditional trappings of a sophisticated intelligent agent, which include *sensors* and *effectors* (Russel & Norvig 1995), *desires* (or values) and *goals* (Bratman, 1987; Pollock, 1995; Schank and Abelson, 1977), *situation analysis* (Kolodner, 1993; Pollock, 1995), *context sensitivity* or *situativity* (Maes, 1990), *efficient resource-bounded cognition* (Kolodner, 1983; Pollock, 1995), and *opportunism* (Hammond 1989; Hayes-Roth & Hayes-Roth, 1979; Simina & Kolodner, 1995). In this context, we will focus on an agent's memory for past experiences and how relevant pieces of multiple past experiences can be effectively integrated during problem solving in order to synthesize solutions for new problems.

2.2. The Need to Combine Experiences

Taking advantage of past experiences is the foundation of case-based reasoning. When confronted with a problem, a case-based reasoner recalls a past experience and adapts it to provide the solution to the new problem. Unfortunately, in many real-world domains we cannot count on a single past experience to provide the outline of a solution to our problems. For example:

- A graduate student asked to present his first paper at an overseas conference must draw on separate past experiences in preparing talks for conferences within his country and preparing his passport and flight arrangements for vacations outside of his country.
- A host planning his first large dinner party must recall both the outline of a menu as served at family gatherings and his separate experiences at preparing individual dishes for himself.
- A home hobbyist attempting his first large piece of furniture must recall both past examples of that type of furniture to provide a design and

experiences with acquiring, assembling and finishing individual components.

All of these problems have something in common: every *piece* of the solution can be constructed entirely out of the agent's past experience (with suitable adaptation), but no *single* past experience suffices to solve the entire problem. For these types of problems, unless a case-based reasoning system has the ability to combine several past experiences, it will have to resort to expensive from-scratch reasoning in order to solve the problem.

Some CBR planning systems combine multiple cases during reasoning. However, they either gather all partial plans at retrieval prior to adaptation (e.g., PRODIGY/ANALOGY, Veloso 1994; Chapter 8, this volume), break plans into *snippets* at storage time so they can be retrieved individually (e.g., CELIA, Redmond 1990, 1992), or combine cases recursively, applying complete past cases to sub-problems within a larger problem (e.g., ROUTER, Goel et al., 1994; SBR, Turner, 1989; PRODIGY/ANALOGY, Veloso, 1994; Chapter 8, this volume). None of these approaches is entirely satisfactory, for various reasons.

It is not entirely clear that all of the knowledge needed to solve a problem can be assembled at the beginning of problem solving. For example, in the furniture example, it is not entirely clear whether or not the agent needs to buy new sandpaper, and hence unclear whether the agent should recall past experiences of buying sandpaper at a hardware store. This uncertainty arises out of several concerns: the uncertainty of the world state (how much sandpaper does the agent have?), uncertainty in the effectiveness of agent actions (how much wood will a piece of sandpaper sand?) and the potential of exogenous events that can invalidate parts of the plan (if a friend drops and scars a piece of the furniture, will the agent have enough sandpaper to remove the scar, or will he need to buy more?). But it can also arise out of *the plan itself:* until the agent has decided on a design for the piece and how much wood will be involved, it is unclear precisely how much sandpaper is needed, and hence unclear whether or not a plan should be retrieved. If some amount of sandpaper is on hand, the goal of acquiring sandpaper may not even arise until late in the planning process, when it has become clear that the amount on hand is insufficient.

Precomputing case snippets also has drawbacks. While this allows us to extract subparts of a case to meet the needs of an open goal in a plan, these subparts need to be computed at storage time. Unfortunately, it is not clear that every useful breakdown of a case can be computed in advance. For example, in the foreign conference example, the two past experiences must be closely interleaved to produce a new plan. If the agent has not stored the passport experience as a separate snippet, the agent may not be able to extract that particular piece of a case if it is retrieved — if the agent was able to retrieve it at all.

Recursive case-based reasoning — using case-based reasoning to satisfy subgoals in a partial plan — is an effective way to combine multiple cases because information about the partially constructed solution can be used to help select additional cases to complete the solution. However, existing systems that use recursive case-based reasoning, such as ROUTER and PRODIGY/ANALOGY, can only retrieve whole cases and attempt to apply them to the subproblems at hand; these systems tend to fill in one 'gap' at a time and neither search for nor attempt to use cases that might fill several gaps in the plan at once, even if such cases exist.

We advocate a more flexible approach: using more complete information about the *current* state of a partially adapted plan to guide the retrieval of cases which address as many deficiencies in

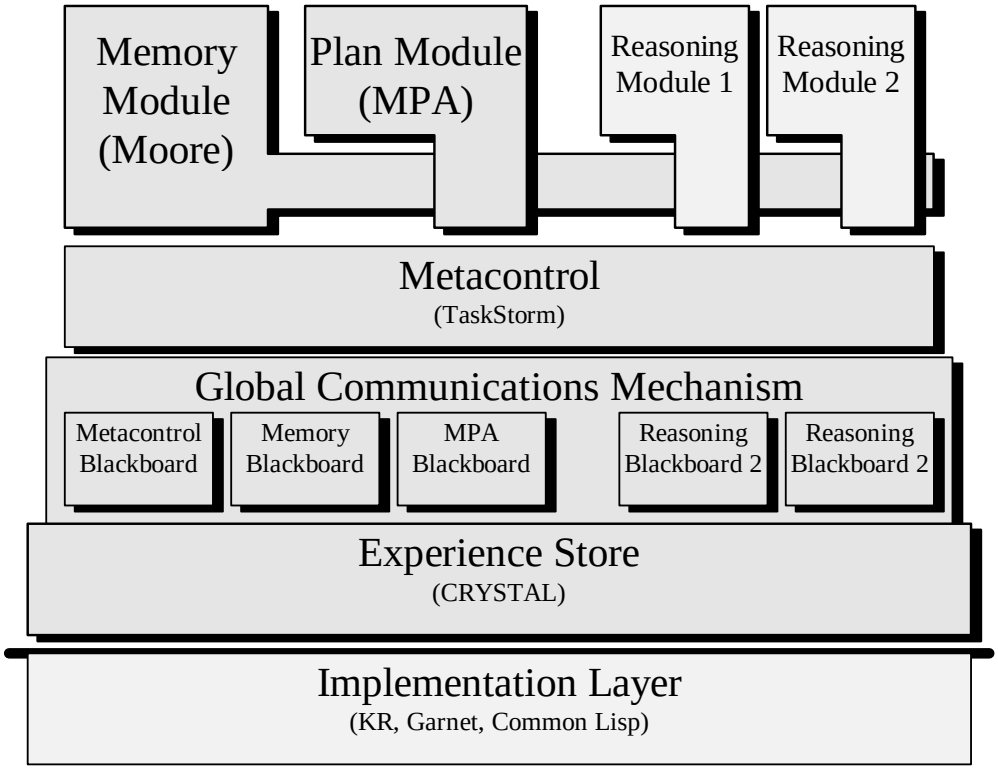

Figure 1. The Architecture of NICOLE.

the plan as possible, and dynamically selecting the relevant portions of those cases to integrate with our current reasoning process. We believe that the key to achieving this is not to attempt to solve this problem in isolation, but instead to look at how the design of a complete agent could provide us with the tools with which to solve the problem.

3. An Architecture for Real-World Domains

3.1. Experience-Based Agency

A growing community of AI researchers has come to believe that successfully meeting the challenges posed by the real world will require building comprehensive agent architectures, rather than by tackling individual problems separately and trying to combine the solutions after the fact (e.g., Anderson, 1983; Hayes-Roth, 1995; Newell, 1990; Nilsson, 1995; Pollock, 1995). We subscribe to this view; in particular, we believe that the problem of deciding when and what cases to retrieve and how to integrate those cases into the system's current plans can only be solved in the context of the retrieval and reasoning needs of an entire agent functioning in a real-world domain.

To explore this problem, we have developed a theory of *experience-based agency,* which specifies a class of agent capable of not only integrating experiences on demand but also retrieving those experiences in a context-sensitive and asynchronous fashion. The theory has five primary components. The core components are a richly represented *experience store* and a *global communications mechanism*, which together lay the foundation for a *context-sensitive asychronous memory system*. To cope with the potential retrieval of new information at any time,

reasoning mechanisms are equipped with *integration mechanisms*; to coordinate reasoning and memory, the theory proposes a central *metacontroller*. Figure 1 illustrates our current implementation of these components in the NICOLE[1] multistrategy reasoning system.

3.2. Memory and Control in an Experience-Based Agent

An experience-based agent is designed to operate in a dynamic, unpredictable world with limited information, and as such naturally requires the ability to combine multiple experiences on demand, clip out their irrelevant subparts, and splice them together into a complete solution for the problem at hand. Driving this integration of experience is the asynchronous retrieval of relevant experiences by the independent memory system. In a dynamic world, we cannot guarantee that the cues and specifications that reasoning provides to memory will be sufficient to allow retrieval of the best experiences quickly enough to allow retrieval to continue uninterrupted; an experience-based agent avoids this problem by allowing memory to return a "best guess" initially while continuing to search memory in parallel with any reasoning, acting, or sensing operations being performed by the agent.[2] Figure 2 illustrates the "life history" of a typical parallel memory search.

The "rich" knowledge representation used in an experience-based agency system's experience store (implemented in NICOLE as a highly connected semantic network[3]) allows the memory to make connections between reasoning operations and past cases; the global communications mechanism (implemented in NICOLE as a set of task-specific blackboards) ensures that the content of reasoning operations is visible to the memory to make it context-sensitive. In contrast, where knowledge representation and the global communications mechanism aim to increase memory's ability to retrieve, the metacontroller limits it: it provides utility metrics which limit when memory (or other reasoning modules[4]) can execute. If no sufficiently suitable retrieval can be found, metacontrol ensures that the agent spends its time processing the information it already has, rather than allow the memory to return every partial match, no matter how slight.

3.3. Reasoning in an Experience-Based Agent

However, while these components are necessary to achieve the desired behavior, they are not sufficient; modification must be made *within* reasoning and memory as well. Traditional reasoning algorithms have well-defined inputs and outputs, and it simply does not make any sense to say that "new information can be retrieved at any time" without some *integration*

---

[1] NICOLE is named after a sentient computer in a science fiction short story by one of the authors (Francis 1995).

[2] Because of its ability to return a best guess, an experience-based agent has many similarities to the interactive "shoot-first" case-based reasoning systems advocated by Riesbeck (1993). Given a problem input by the user, a shoot-first system attempts to quickly retrieve an approximate set of cases and/or propose a sketchy solution, and then uses feedback from the user to redefine the problem, refine the search, or adapt the solution. Both experience-based agents and shoot-first systems require similar memory capabilities; however, the primary "user" of an experience-based agent's quick retrievals is the agent itself.

[3] This semantic network is "highly connected" in two senses: both in the network structure (as in Kilmesch, 1994) and in the semantics of the network, in which all nodes and links are reified concepts (as in Brachman, 1985; Wilensky, 1986).

[4] Reasoning modules are implemented in NICOLE as supertasks (Moorman & Ram, 1994) which combine characteristics of task-specific blackboard panes (Hayes-Roth & Hayes-Roth, 1979) and extended Reactive Action Packages (RAPs; Firby, 1989).

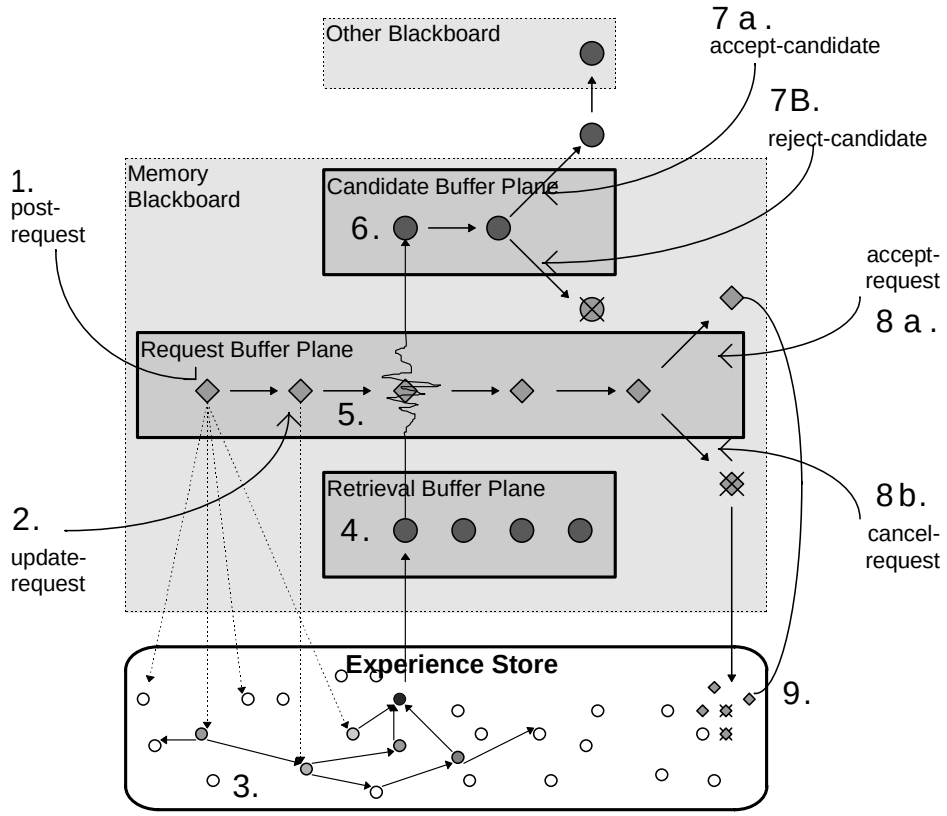

Figure 2. The Life History of a Retrieval in NICOLE.

1. A retrieval begins when a reasoning module calls **post-request**, which adds a new **request-node** (symbolized by a diamond) to the **request buffer** of the memory blackboard. **2.** A request-node may be updated with a **update-request** call at any time. **3.** Each time a request is posted or updated (or when activity occurs in other system blackboards) **activation** spreads to nodes in long term memory (the **experience store**). **4.** On every retrieval cycle, a limited number of **active nodes** (symbolized by dark circles) are retrieved to the **retrieval buffer** for consideration. **5.** Each pending request in the request buffer is **matched** against the candidates in the retrieval buffer. **6.** Successful matches are posted to the **candidate buffer** to be copied to the requesting blackboard or process. **7.** A retrieval candidate can be accepted or rejected by the **accept-candidate** and **reject-candidate** calls, which update the state of the request to allow it to more accurately select future matches. **8.** The reasoning module may at any time decide to terminate processing of a request by accepting it through an **accept-request** call or rejecting it with a **cancel-request** call. **9.** Terminated requests, both successful and unsuccessful, are stored in long term memory and used by the **storage module** (not shown) to adjust associative links in long term memory and retrieval parameters in the matching and candidate retrieval systems.

mechanism that can incorporate that information into reasoning. Similarly, traditional retrieval algorithms operate at the command of the reasoning system, performing each search of memory separately from every other and basing each search only on the cues and specifications provided at the particular moment the search was initiated. In order to take maximum advantage of an independent memory we must allow it to establish *knowledge goals* which are based on the needs of reasoning but which can be independently pursued, updated, and (eventually) satisfied.

3.4. Exploiting Past Plans in an Experience-Based Agent

In addition to the general need to combine past experiences into a coherent solution to its current problems, if an experience-based agent is called upon to perform planning tasks, it needs the specific capability to combine multiple *plans* on demand, clip out their irrelevant subparts, and splice them together into a complete plan. To solve this problem, we have configured NICOLE to implement an interactive multi-plan adaptation system (called NICOLE-MPA).

NICOLE-MPA represents an advance over traditional case-based reasoning systems in two ways. First, NICOLE-MPA uses a novel algorithm called the Multi-Plan Adaptor (MPA) to extend the concept of multi-plan adaptation to the least-commitment case-based planning framework. Within the context of the NICOLE system, the MPA algorithm also provides both an integration mechanism and a knowledge goal generation mechanism for least-commitment planners using partially ordered plans. Second, NICOLE-MPA provides a framework for the study of various heuristics for multi-plan adaptation.

To set the stage for NICOLE-MPA, we will discuss the least-commitment planning framework and its case-based implementation in systems like SPA, then detail the MPA algorithm itself and how it extends the traditional systems upon which it is founded. Then, we will discuss how NICOLE is configured to implement interactive multi-plan adaptation, and conclude by discussing the potential efficiency gains and hazards of NICOLE-MPA.

4. Interactive Multi-Plan Adaptation

4.1. An Overview of Case-Based Least Commitment Planning

Least-commitment planning departs from traditional planning systems by delaying decisions about step orderings and bindings as much as possible to prevent backtracking (Weld, 1994). Least-commitment planners solve problems by successive *refinement* of a partial plan derived from the initial and goal conditions of the problem. Plans are represented as sets of steps, causal links between steps, variable bindings and ordering constraints. Beginning with a skeletal partial plan based on the initial and goal conditions of the problem, a least-commitment planner attempts to refine the plan by adding steps, links and constraints that eliminate open conditions or resolve threats.

The Systematic Plan Adaptor algorithm (Hanks & Weld, 1995) is a least-commitment algorithm for case-based planning. SPA is based on four key ideas: treat adaptation as a *refinement* process (based on the generative least-commitment planner SNLP; McAllester & Rosenblitt, 1991), annotate partial plans with *reasons* for decisions, add a *retraction mechanism* to remove decisions, and add a *fitting mechanism* to fit previous plans to current situations. Reasons support the fitting and retraction mechanisms by allowing SPA to determine the dependencies between steps. Once a plan has been retrieved, it may contain steps and constraints that are extraneous or inconsistent with the new situations; during fitting, reasons allow SPA to identify extraneous steps and remove them; during adaptation, reasons allow SPA to isolate steps which are candidates for retraction.

One limitation of SPA is that it is a single-plan adaptor; even if a new problem could be solved by merging several plans, SPA must choose only one and adapt it to fit. This limitation arises from SPA's attempt to maintain *systematicity* and *completeness*. A systematic planner never

repeats its work by considering a partial plan more than once; a complete planner always finds a solution if it exists. SPA ensures these properties (in part) by choosing only one refinement at a time, by never retracting any refinement made during adaptation (to avoid reconsidering plans), and by adding all possible versions of any refinement it chooses (to avoid missing a solution). A full discussion of completeness and systematicity in SPA is beyond the scope of this paper (but see Francis & Ram, 1995; Hanks & Weld, 1995).

4.2. Multi-Plan Adaptation

Because it adapts only one plan, SPA can resort to significant amounts of from-scratch planning even when the knowledge needed to complete the plan is present in the case library. To make the most effective use of the planner's past experience, we need the ability to recognize when a partial plan needs to be extended, select plans to that address the deficiency, and then extract and merge the relevant parts of the retrieved plan into the original plan.

MPA resolves this problem in SPA by allowing the retrieval and merging of arbitrary numbers of cases at any point during the adaptation process. MPA also allows the dynamic extraction of relevant parts of past cases. To achieve this, we extended the SPA framework by adding three key mechanisms:

- a goal deriver, which extracts *intermediate goal statements* from partial plans
- a *plan clipper,* which prepares plans for merging using a modified plan fitting mechanism
- a *plan splicer,* which merges two plans together based on their causal structure

Intermediate goal statements are MPA's knowledge goals; they provide MPA with the ability to merge partial plans at any point of the adaptation process and contribute to its ability to dynamically extract the relevant subparts of retrieved cases. Intermediate goal statements are extracted by the inverse of a representational "trick" that SPA uses to construct a skeletal plan if it can't find a relevant case in its library. The trick is simple: build a skeletal plan by adding dummy initial and final steps whose post- and pre-conditions match the initial and goal conditions of the problem. This technique is also used in SPA's generative predecessor SNLP (McAllester & Rosenblitt, 1991) as well as a host of other STRIPS-based planning systems. MPA inverts this trick by extracting new goals from the open conditions of a partial plan. As planning proceeds, open conditions in the goal statement are resolved, but new open conditions are posted as new steps are added. MPA constructs an intermediate goal statement by extracting these new open conditions and using them to form the new goal state, and by extracting the initial conditions of the partial plan can be extracted and using them to form the new initial conditions.[5]

---

[5] Unfortunately, since ordering constraints and binding constraints may be posted to the plan at any time, only the initial conditions can be guaranteed to be valid conditions for the intermediate goal statement. Conditions established by other steps of the plan may be clobbered by the addition of new steps and new ordering constraints. However, it might be possible to develop heuristics that select additional initial conditions that are likely to hold, perhaps in conjunction with more complex retrieval, fitting and splicing algorithms. In general, deciding which parts of a plan can be extracted to form a sensible and effective intermediate goal statement is a difficult and unsolved problem.

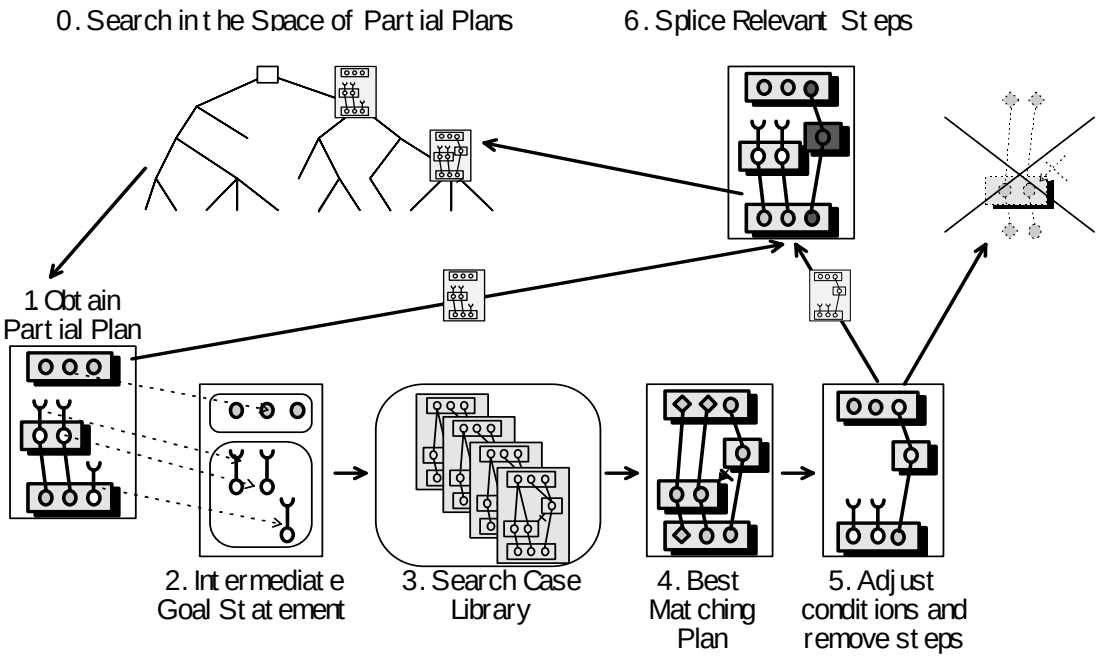

Figure 3. Overview of Multi-Plan Adaptation

Multi-plan adaptation begins when **1**. a partial plan is obtained, either directly from a goal statement, from an initial case fitting, or from ongoing adaptation processes. **2**. An intermediate goal statement is extracted from the plan, consisting of the initial conditions known to be true in the world and the set of preconditions not yet satisfied in the plan. **3**. The case library is searched for a matching plan in exactly the same way that it is searched for initial case fittings. **4**. The best matching plan is retrieved and **5**. is adjusted to have the right set of initial and goal conditions and to remove extraneous plan steps. **6**. The steps are recursively spliced into the plan, beginning with the links that match to the intermediate goal statement and then moving backwards through the plan along the paths of the causal links. **0**. Finally, the successfully spliced plans are returned for further adaptation or splicing.

Just like the original goal statement, the intermediate goal statement can be used to retrieve and fit a partial plan. However, the result of this process is not a complete fitted plan suitable for adaptation; it is a *plan clipping* that satisfies some or all of the open conditions of the partial plan from which the intermediate goal statement was derived. To take advantage of the plan clipping for adaptation, it must be *spliced* into the original partial plan. Together, plan clipping and splicing form MPA's integration mechanism for incorporating new plans into the current reasoning context (Figure 3).

Our splicing mechanism uses the intermediate goal statement to produce a mapping between the partial plan and the plan clipping, pairing open conditions from the partial plan with satisfied goal conditions from the plan clipping. The plan splicer uses this mapping to perform a guided refinement of the original partial plan, selecting goal conditions from the clipping and using the links and steps that satisfied them as templates to instantiate similar steps and links in the original plan. As these steps are added, new mappings are established between open conditions in the new steps and satisfied preconditions in the clipping and are added to the queue of mappings that the splicer is processing. Hence, the plan splicer performs a backwards breadth-first search through the causal structure of the plan clipping, using links and steps in the clipping to guide the instantiation of links and steps in the original plan. Figure 4 briefly outlines the MPA algorithm.

Input: A partial plan P, and a case library C.
Output: A new partial plan P'.

procedure **MPA** (P, C):
1 P' ← **Copy-Plan**(P)
2. igs ← **GetIntermediateGoalStatement**(P')
3. plan ← **RetrieveBestPlan** (C, igs)
4. {clipping, mapping} ← **FitPlan** (plan, igs)
5. for cgp in mapping do
6. if **Producer-Exists** (oc-gl-pair, P')
7. then **Splice-Link** (oc-gl-pair, P', clipping)
8. else **Splice-Step** (oc-gl-pair, P', clipping)
9. **AddNewOpenCond-GoalPairs**(mapping, P')
10. return P'

Figure 4. The MPA Algorithm

4.3. Controlling Multi-Plan Adaptation

Merely having the ability to splice plans together does not allow us to take advantage of past experience. We need to decide what experiences to combine and when to combine them. Because the MPA algorithm can potentially be performed at any point during the adaptation process — using an initial skeletal plan derived from the initial and goal statement, using a fitted plan derived from retrieval, or using an adapted plan after some arbitrary amount of retraction and refinement — we have considerable flexibility in deciding what to retrieve, when to retrieve it and when to merge it.

We have considered three alternative control regimes, each of which makes different commitments about when to retrieve and when to adapt. On one end of the spectrum, *Systematic MPA* preserves SPA's property of systematicity by splicing all retrieved cases before (generative) adaptation begins. On the other, *Extreme MPA* never performs generative adaptation and instead uses a set of *pivotal cases* (Smyth & Keane, 1995) to guarantee completeness.

Both Systematic and Extreme MPA make extreme commitments: either integrate all knowledge before generative adaptation begins, or never generatively adapt and rely solely on past experience. An alternative approach is to allow plan splicing at any point during adaptation. In the middle stands *Interactive MPA*, a control regime that can potentially attempt a retrieval at any time, either with the initial skeletal plan or with partial plans produced as a result of adaptation. Since the results of splicing cause large jumps in the search space, this regime deliberately departs from the systematicity of SNLP and SPA in an attempt to solve the problem with less search.

However, allowing arbitrary plan retrieval and plan splicing is not without cost. Performing a full search of the system's case library at every step of the problem space could be computationally prohibitive. The costs of searching the case library at every step of the problem space could outweigh the benefits of reduced search, especially if the system enters a "slump" — an adaptation episode which begins and ends with the application of relevant clippings, but which goes through a series of intermediate plans for which the system cannot match any existing plans in its case library. Clearly, it is worthwhile to retrieve and apply clippings at the beginning and

end of a slump, but a full search of the case library at each intermediate step could cost more than the benefits that the initial and final retrievals provide. This is the *swamping utility problem* — the benefits of case retrieval can be outweighed by the costs of that retrieval, leading to an overall degradation in performance as a result of case learning (Francis & Ram, 1995).

4.4. Interactive Multi-Plan Adaptation

Developing heuristics for deciding when and when not to retrieve is a challenging open problem. The experience-based agency theory suggests that plan adaptation should be driven by retrieval; to test this theory, we have implemented an Interactive MPA system within a specially configured version of NICOLE, called NICOLE-MPA.

NICOLE-MPA is an instantiation of NICOLE in which all of MPA's temporary data structures are implemented using NICOLE blackboards. Case adaptation, intermediate goal statement generation, plan clipping, and plan splicing are all implented as NICOLE task modules, which run in parallel with the memory module and other modules of the NICOLE system. Because of this parallelism, NICOLE-MPA can potentially attempt a retrieval at any time, either with the initial skeletal plan or with partial plans produced as a result of adaptation. To support this retrieval, goals, plans and intermediate goal statements are augmented with *plan tags*, which allow the smooth integration of traditional least-commitment planning structures with the richly interconnected semantic network which makes up NICOLE-MPA'S experience store. The plan tagging mechanism allows NICOLE-MPA to manipulate intermediate goal statements, plans and goals in a completely native fashion while providing the memory system with the cues necessary to continue to refine retrieval.

A typical problem-solving session in NICOLE-MPA begins with the presentation of a problem. From this problem, NICOLE-MPA generates a goal statement and uses the goal statement to post a retrieval request. If some partially matching case can be found, the system returns it as its *current* best guess; however, the retrieval request remains active. As the system adapts the plan (using a specially "wrapped" and modified version of the SPA algorithm incorporated into a NICOLE task module), it generates plan tags for each partial plan it generates, providing additional cues for the memory module to attempt further retrievals. When memory finds a new past case whose degree of match exceeds a certain threshold, NICOLE-MPA's metacontroller allows memory to return it as a new guess. If it is applicable, the metacontroller may schedule the plan splicing module, which will integrate it into the current partial plan.

4.5. The Benefits of Multi-Plan Adaptation

Both adapting a single partial plan and adapting merged partial plans can produce significant benefits over generative problem solving. The cost of generative planning is exponential in the size of the final plan produced, whereas fitting a plan is a linear operation in the size of the plan. Hence, the potential exists for substantial improvement through retrieval and adaptation if an appropriate past plan exists, especially for large plans. In certain domains, SPA has demonstrated significant improvements over generative planning. However, if large gaps exist in the retrieved partial plan, SPA must resort to adaptation, which, like generative problem solving, has an exponential cost in the number of steps that need to be added.

While this amount of adaptation may be a significant improvement over complete from-scratch problem solving, the potential exists to reduce that even further by using MPA to clip and splice more partial plans. Fitting a clipping and splicing a clipping are linear operations in the size of

the plan being spliced. Hence, the potential exists for substantial improvement through plan merging if an appropriate past plan exists, especially if the gaps in the existing plan are large. An initial implementation of MPA for a test domain indicated significant speedups (beginning at 30%) over SPA for even the smallest examples (solution size of the final plan = 5 steps).

5. Related Work

There are wide bodies of work on both least-commitment planning and case-based reasoning. The most relevant example of that work to this research is of course SPA, upon which MPA builds. Other similar plan reuse systems include PRIAR (Kambhampati & Hendler 1992) an SPA-like system based on NONLIN, and XII (Golden et al 1994), an SPA-like system that plans with incomplete information. Hanks and Weld (1995) discuss these and other plan reuse systems from the perspective of the SPA framework.

MPA's plan splicing mechanism is in many ways similar to DERSNLP (Ihrig & Kambhampati 1994), a derivational analogy system built on top of SNLP that uses *eager replay* to guide a partial order planner. While DERSNLP's eager replay mechanism is in some ways similar to a limiting case of Systematic MPA in which a single plan is retrieved and spliced into a skeletal plan derived from an initial problem statement, DERSNLP goes beyond SPA's reason mechanism and includes a full derivational trace of problem solving in its cases. While DERSNLP and its extension DERSNLP-EBL focus on when it is profitable to retrieve a partial plan, unlike NICOLE-MPA they do not provide the capability of interrupting adaptation as a result of an asynchronous memory retrieval, nor do they provide the ability to integrate the results of multiple plans.

Combining multiple plans in case-based reasoning is not a new idea. The PRODIGY/ANALOGY system (Veloso 1994; Chapter 8, this volume) can retrieve and merge the results of an arbitrary number of totally ordered plans during the derivational analogy process. However, because PRODIGY/ANALOGY manipulates and stores totally ordered plans, it runs into significant issues on deciding how to interleave steps (Veloso, 1994, p124-127), an issue MPA avoids because of its least-commitment heritage. Furthermore, PRODIGY/ANALOGY deliberately limits its capability to retrieve and combine cases on the fly in an attempt to reduce retrieval costs.

The ROUTER path-planning system (Goel et al., 1994) has the ability to recursively call its case-based methods to fill in gaps in a planned route when no exactly matching case can be found. (PRODIGY/ANALOGY has a similar capability to call itself recursively, although the full implications of this ability have not yet been explored). Like NICOLE-MPA, ROUTER has the ability to combine multiple cases, although only in a synchronous fashion because its memory does not support spontaneous retrieval. However, each of these cases must be complete cases; it does not have the ability to clip out the relevant portion of a case at retrieval time. ROUTER does have the ability to break a case up into subcases at storage time, but the results show that computational costs of this storage computation outweigh the benefits in improved retrievals (Goel et al. 1994, p. 63).

The JULIA system (Hinrichs 1992) also has the ability to combine pieces of several past cases, but this is largely a domain-dependent algorithm for merging declarative structures, rather than a domain independent planning system. The CELIA system (Redmond 1990, 1992) stores cases as separate *snippets*, case subcomponents organized around a single goal or set of conjunctive goals.

Snippets provide CELIA with the ability to retrieve and identify relevant subparts of a past case based on the system's current goals. Note that while snippets are superficially similar to plan clippings, plan clippings are constructed dynamically during problem solving, whereas snippets need to be computed and stored in advance.

Clippings are similar to macro operators (Fikes et al. 1972) in that they use past experience to combine several problem solving steps into a single structure that can be applied as a unit, allowing the system to make large jumps in the problem space and avoid unnecessary search. However, macro operators differ from clippings in two important ways. First, macro operators are traditionally precomputed at storage time, whereas clippings are computed dynamically; second, macro operators are fixed sequences of operators, whereas clippings are partially ordered sets of operators that may be resolved in a wide variety of ways in the final plan.

Kambhampati & Chen (1993) built and compared several systems that retrieve partially ordered case-like "macro operators". They demonstrated that least-commitment planners could take greater advantage of past experience than totally-ordered planners because of their ability to efficiently interleave new steps into these "macro-operators" during planning. While this work focuses primarily on interleaving new steps into single past plans, the explanations the authors advance for the efficiency gains they detected could be extended to suggest that least-commitment planners would be superior to totally-ordered planners when interleaving multiple plans. The NICOLE-MPA system should provide us a testbed with which we can empirically evaluate this hypothesis.

6. Conclusion

We have presented the Multi-Plan Adaptor, an algorithm that allows a case-based least-commitment planner to take advantage of the benefits of several past experiences. MPA provides the ability to retrieve and merge dynamically selected case components at any point during the adaptation process by extracting an intermediate goal statements from a partial plan, using the intermediate goal statement to retrieve and clip a past plan to the partial plan, and then splicing the clipping into the original partial plan.

Multi-plan adaptation has the potential for substantial speedup over single-plan adaptation, but in order for those benefits to be realized MPA must be embedded within a control regime that decides when the system attempts a retrieval, when the system merges, and when the system resorts to adaptation. We have used the NICOLE multistrategy reasoning system to implement an interactive control regime in which cases may be retrieved at any point during adaptation. To cope with the potentially swamping cost of retrieval at every adaptation step, NICOLE-MPA employs an asychronous, resource-bounded memory module that retrieves a "best guess" and then continues to monitor the progress of adaptation, returning a new or better retrieval as soon as it is found.

We believe that the ability to combine multiple plans and the ability to perform asynchronous retrievals form integral parts of any complete agent that functions in a complex domain; moreover, actually implementing these capabilities in an efficient and sensible way can only be done by considering the architecture of a complete agent. Combined with a metacontroller sufficient to make them work together, multi-plan adaptation and asynchronous retrieval form the cornerstone of our theory of experience-based agency, a theory of how an agent could take maximum advantage of its past experiences to cope with the problems of the real world.

References

Anderson, John R. (1983). *The Architecture of Cognition.* Cambridge, Massachusetts: Harvard University Press.

Brachman, R. J. (1985). An Overview of the KL-ONE Knowledge Representation System. Cognitive Science, 9, (1985) 171-216

Bratman, M. (1987). *Intentions, Plans and Practical Reason*. Harvard.

Fikes, Hart and Nilsson. (1972). STRIPS. *Artificial Intelligence*, 1972, pp. 189-208.

Firby, J. (1989). Adaptive Execution in Complex Dynamic Worlds Ph.D. Thesis, Yale University Technical Report, YALEU/CSD/RR #672, January 1989.

Francis, A. and Ram, A. (1995). A Comparative Utility Analysis of Case-Based Reasoning and Control-Rule Learning Systems. In *Proceedings, ECML-95*, Heraklion, Crete, 1995.

Francis, A.G. (1995). "Sibling Rivalry." *The Leading Edge: Magazine of Science Fiction and Fantasy*, 30, February 1995, pp. 79-102.

Goel, A.K., Ali, K.S., Donnellan, M.W., Garza, A.G., and Callantine, T.J. (1994). Multistrategy Adaptive Path Planning. *IEEE Expert (9) 6*, December 1994, pp. 57-65.

Golden, K., Etzioni, O., & Weld, D. (1994). Omnipotence Without Omniscience: Sensor Management in Planning. In *Proceedings of AAAI-94*. AAAI Press/MIT Press.

Hammond, K. (1989). Opportunistic Memory. In *Proceedings of the 1989 Meeting of the International Joint Committee on Artificial Intelligence.*

Hanks, S., & Weld, D. (1995). A Domain-Independent Algorithm for Plan Adaptation. *Journal of Artificial Intelligence Research 2*, pp. 319-360.

Hayes-Roth, B. (1995). Agents on stage: Advancing the State of the Art of AI. *Knowledge Systems Laboratory Report No. KSL 95-50*. May 1995, Knowledge Systems Laboratory, Stanford University.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science (2)*, pp. 275-310.

Hinrichs, T.R. (1992). *Problem Solving in Open Worlds: A Case Study in Design*. Lawrence Erlbaum.

Ihrig, L. & Kambhampati, S. (1994). Derivation Replay for Partial-Order Planning. In *Proceedings of AAAI-94*. AAAI Press/MIT Press.

Kambhampati, S. & Chen, J. (1993). Relative Utility of EBG based Plan Reuse in Partial Ordering vs. Total Ordering Planning. In *Proceedings of AAAI-93*. AAAI Press/MIT Press.

Kambhampati, S. & Hendler, J. (1992). A Validation Structure Based Theory of Plan Modification and Reuse. *Artificial Intelligence, 55*, 193-258.

Klimesch, W. J. (1994). *The structure of long-term memory: A connectivity model of semantic processing*. LEA.

Kolodner, J.L. (1993). *Case-based Reasoning.* Morgan Kaufmann.

Maes, P. (1990). Situated agents can have goals. In. P. Maes, Ed., *Designing Autonomous Agents: Theory and Practice from Biology and Engineering and Back.* MIT.

McAllester, D., & Rosenblitt, D. (1991). Systematic Nonlinear Planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 634-639. AAAI Press/MIT Press.

Moorman, K. & Ram, A. (1994). Integrating Creativity and Reading: A Functional Approach. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, Atlanta, GA, August 1994.

Newell, A. (1990). *Unified theories of cognition.* Harvard.

Nilsson, N. (1995). Eye on the prize. *AI Magazine (16) 2*, Summer 1995, pp.9-17.

Orasanu, J. & Connolly, T. (1993). The Reinvention of Decision Making. In Klein, G. A., Orasanu, J., Calderwood, R., and Zsambok, C.E., eds., *Decision Making in Action: Models and Methods*. Ablex.

Pollock, J.L. (1995). *Cognitive Carpentry: A blueprint for how to build a person*. MIT Press.

Redmond, M. (1990). Distributed Cases for Case-Based Reasoning: Facilitating Use of Multiple Cases. In *Proceedings of AAAI-90*. AAAI Press/MIT Press.

Redmond, M. (1992). Learning by Observing and Understanding Expert Problem Solving. *Georgia Institute of Technology, College of Computing Technical Report no. GIT-CC-92/43*. Atlanta, Georgia.

Riesbeck, C. (1993) Replacing CBR: Now What? Invited talk. *AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC, July.

Russel, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Morgan Kaufmann.

Schank, R. & Abelson, R. (1977). *Scripts, plans, goals and understanding*. LEA.

Simina, M. & Kolodner, J. (1995). Opportunistic Reasoning: A Design Perspective. In *Proceedings of the 17th Annual Cognitive Science Conference*. Pittsburg, PA, July 1995.

Smyth, B. & Keane, M. (1995). Remembering to Forget: A Competence-Preserving Deletion Policy for CBR Systems. In *Proceedings of IJCAI-95*.

Sternberg, R. J. (1985). Teaching Critical Thinking: Are We Making Critical Mistakes? *Phi Delta Kappa*, November 1985, p194-198.
Sternberg, R. J. (1986). *Intelligence Applied: Understanding and Increasing Your Intellectual Skills*. Harcourt, Brace, and Jovonovitch.
Turner, R. (1989). A schema-based model of adaptive problem solving. Ph.D. diss., *GIT-ICS-89/42.*, Department of Information and Computer Science, Georgia Tech.
Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Springer-Verlag.
Weld, D. (1994). An introduction to least-commitment planning. *AI Magazine, (15) 4*, Winter 1994, pages 27-61.
86/294 Dept. of Electrical Engineering & Computer Science, University of California - Berkeley.
Wilensky, R (1986). Some Problems and Proposals for Knowledge Representation *Technical Report #UCB/CSD 86/294* Dept. of Electrical Engineering & Computer Science, University of California - Berkeley.
Wooldridge, M. & Jennings, N. (1995) Intelligent Agents: Theory and Practice. Submitted to *Knowledge Engineering Review* October 1994, Revised January 1995.