COMBINING WEAK LEARNING HEURISTICS IN GENERAL PROBLEM SOLVERS

T.L. McCluskey

The City University, Northampton Square,
London, England.

ABSTRACT

This paper is concerned with state space problem solvers that achieve generality by learning strong heuristics through experience in a particular domain. We specifically consider two ways of learning by analysing past solutions that can improve future problem solving: creating macros and the chunks. A method of learning search heuristics is specified which is related to 'chunking' but which complements the use of macros within a goal directed system. An example of the creation and combined use of macros and chunks, taken from an implemented system, is described.

I INTRODUCTION

Integrating ideas and techniques devoloped in Machine Learning, with those of Problem Solving, has attracted substantial recent research effort (e.g. Laird et al 86, Korf 85, Langley 85, Mitchell et al 83). An important aspect is the revival of the 'general' problem solver. Its demise was due in part to the failure of its weak heuristics to tackle problems of complexity in some given application domain; now it returns equipped with not just weak problem solving heuristics but with weak heuristics for learning strong, i.e.domain dependent, heuristics. The latter may take the form of useful shifts in the problem space representation (a simple example is the learning of macro operators) or improving search through a particular space by the acquisition of search control heuristics. Thus, while its generality is maintained, learning may improve the problem solver's efficiency during the application to a particular domain. This is the approach we have taken in the construstion of a 'heuristic learning problem solver shell' called FM; it can acquire strong heuristics from problem solving experience when it is applied to specific domains. A complementary approach is to acquire or discover them during a preprocessing stage as in [Iba 85], [Korf 85]) and [Dawson & Siklossy 77].

FM's application domains can have variable initial and goal states. Applications are interchangeable by specifying domain environments, states and goals as expressions in first order logic, and operators in terms of structured add, delete and precondition predicates. Control stategies may be Interchanged (e.g. forward best-first or goal reduction) as can weak learning methods such as macro and chunk creation.

This constitutes a more general approach to recent work on heuristic learning in problem solvers (e.g. [Mitchell et al 83], [Korf 85]), where systems typically improve in domains with a fixed goal, employ a more specialised representation scheme, and a forward state space search strategy. This paper will outline FM's goal directed search and describe how macros and chunks are created and used as complementary heuristics during that search.

II GOAL NODE SEARCH IN FM

The backward search of FM proceeds in a goal reduction manner, starting with the initial goal, through a space of goal nodes (similar to those in [Dawson & Siklossy *77]).* Each goal node can be modelled as a 6-tuple:
(identifier, goal, initial state, ancestors, purpose, trace)
The trace records attempts to solve the goal, whereas the purpose records why the goal node was created (typically to solve the unsatisfied preconditions of an operator). Goals, expressed as conjunctions of ptedicates, are initially assumed to be decomposable: when a goal node is activated, operator instantiations which add goal predicates have their unsatisfied preconditions form another goal node, unless they are already satisfied in which case those operators are applied to the initial state and the result recorded in the trace.

When the trace of a goal node eventually contains a state satisfying its goal (via an operator sequence Os), we say that the goal node is solved, and all nodes which are ancestors of it are removed from the search. If it was activated to solve an operator O's preconditions, then the sequence Os + 0 is applied to the goal node's parent's initial state and the result recorded in the parent's trace.

A goal node's initial state may be the state inherited from a parent node, or may be an advanced state partially satisfying the parent's goal. The latter is the case when goals cannot be solved by simple decompositon; FM examines the trace and forms new goal nodes whose goal predicates are inherited but whose initial states are selected from intermediate states taken from the parent's trace.

The kind of representation of goal nodes outined above aids both the formation and use of strong heuristics. The trace is available for analysis and criticism after the solution of each goal node, allowing 'within-trial transfer of learning' (see [Laird et al 84]) to take place. In our implementation of FM we have experimented with the formation of closed macros, 'b-chunks' and also subgoal ordering heuristics at this stage, but we shall limit our discussion to the first two.

III CLOSED MACRO CREATION

We consider a closed macro operator to be an operator sequence that has been compiled and generalised into a form similar to that of a primitive operator (in contrast to the 'open' macrops of [Fikes et al 72]). This sequence forms part of a past solution, in the case of learning by experience, which includes fully instantiated operators and intermediate states. Here the compilation involves finding the sequence's weakest precondition through the intermediate states and using it as the macro's precondition. Within this certain constants can then be selectively generalised using a technique similar to the Explanation-Based Learning of [Mitchell et al 86].

Systems that learn closed macros ([Minton 85], [Iba 85]) seem to demonstrate significant improvement in problem solving within robot and puzzle worlds but there are pitfalls in using this technique as the sole learning component:

-search trees do shorten but unfortunately grow bushy since distinct instantiations of macros proliferate. (This is reminisent of the effect of paramodulation, a 'macro inference rule' in Theorem Proving, which combines resolution with the axioms of equality, but when used in search changes long thin trees to short bushy ones!).

-solutions which comprise of closed macros are prone to produce non-optimal paths even after checks for redundant primitive operator sequences have been made.

We claim that such problems may be overcome by the learning of strong heuristics such as chunks to complement the use of macros.

Macros are created and stored in FM when goal nodes are solved, and then are immediately available for use in problem solving. Each are compiled from a successful operator sequence into a primitive operator format. The major part of this compilation process is in building up the precondition M.p (a conjunction of predicates) of a macro M. This is accomplished by a procedure modelled on goal regression equations:

M.p = Pn where PO = G and
Pi *m* (Pi-1 - 0[n+1-i].a) U 0[n+1-i].p , i = 1 to n

where 'U' and '--' mean set union and difference, 0[i].p, 0[i].a stand for the precondition and add predicates of operator i respectively, and G the goal predicates for the solution sequence.

Constants that appeared as arbitrary members of some particular type in the solution's operator sequence are carefully generalised to a variable with that type restriction (following [Kodratoff 84]). Generalisation is justified since no operator in the solution sequence referred to the constant specifically but only to its type. Identical constants are generalised to the same variable throughout the macro, but equality binding restrictions are added where variables of the same type are generalised from distinct constants, so that they may not be instantiated to the same constant when in use. Macros are then incorporated into future problem solving as primitive operators, although some may later be deleted if rarely used.

IV B-CHUNK CREATION

The chunks created by FM improve the system's subsequent problem solving behaviour by providing search control knowledge. They are formed during the goal directed search and advise on the search through partial solutions. The absence of such a learning component in STRIPS with Macrops is pointed out in [Porter and Kibler 84] and Minton's Morris system [Minton 85] apparently combines only weak search heuristics with the use of macros.

Consider 0[i] (1<i<n) taken from an operator sequence 0[1],0[2]. . . .0[n] which achieves a goal node (with goal predicate(s) G) from a initial state I within a domain environment E (E is a set of facts and rules constituting background knowledge for a particular application). A b-chunk (0[i]'; G'; P') is built for each 0[i] to the following specification: consider a function 'sim':

sim : CP x CP x CP x NatO --> CP

where CP is the space of conjunctions (or sets) of Predicates and

sim(X,Y,E,O) = {P in Y: P logically follows from X&E}
sim(X,Y,E,N) = sim(X,Y,E,N-1) union
{y el. of Y, e subset of E : y is related to an x in X
by an association chain e of length N}

Then

P= sim(M(i),M(1),E,K) where M(j)= the macro precondition (see section III) of sequence 0[j],0[j+1], ...,O[n];K>=0,

and finally

(0[i]'; G'; P') = the careful generalisation of (0[i]; G; P).

When K = 0 then 0[i]'s chunk's third component may be roughly described as those predicates which were present in the goal node's initial state and that were also involved in the achievement of G after 0[i-1]. This includes environment information (which is assumed to be a part of every state) that has been used in the satisfaction of the operator's preconditions. FM initially forms P with K=0 and then checks to see if the resulting chunk would be discriminatory if used to solve the same goal node again. If it is not the case then K is incremented and P is augmented with predicates using an 'association chain' technique similar to that described in fVere 771.

B-chunks are then used during subsequent search when FM finds multiple operators (or operator instantiations) are available to achieve a goal predicate Gp, but none of their preconditions are completely satisfied. A b-chunk (01; G1; P) will favour an operator instantiation O applied to a goal node if P logically follows from I&E under the variable bindings obtained by the successful matching of 01 to O, and G1 to either Gp or one of Gp's ancestors. The instantiation(s) favoured by the most chunks is then chosen to form a new goal node.

V COMBINED USE OF LEARNT HEURISTICS

To clarify the combined use of closed macros and b-chunks we use a simple example. We applied FM to a robot world using a similar operator set to [Fikes et al 72]. After box moving tasks it forms macros such as:

(name: macro21(Rm1, Dr1,Rm2,Box,Dr2,Rm3),
preconditions: in_room(Box,Rm1)&nexMo(robot,Box)
&connect(Rm1,Rm2,Dr1)&connect(Rm2,Rm3,Dr2) ....
add: in room(Box,Rm3),
side_effects: in_room(robot,Rm3), ...).

Macro21 is equivalent to the primitive sequence:
{pushto(Box,Dr1,Rm1), pushthru(Box,Dr1,Rm2),
pushto(Box,Dr2,Rm2), pushthru(Box,Dr2,Rm3)}.

In solving the goal 'in_room(boxA, room4)' from the situation in figure 1, macro21 constitutes the part of the solution shown by an arrow. One b-chunk (where K=1 in section IV) created to advise on its use is *(note: we leave out some details; capital letters denote variables):*

(macro21(Rm1, Dr1,Rm2,Box,Dr2,Rm3) ;
in_room(Box,Rm3) ;
in_room(Box,Rm4)&connect(Rm4,Rm1,Dr3)&
connect(Rm1,Rm2,Dr1)&connect(Rm2,Rm3,Dr2)& ...)

In a future problem, this chunk will support the inclusion of instantiations of macro21 in partial solutions which conform to its constraints. For instance, consider task in_room(boxB,room6). It can be seen by the description of chunk use in section IV that instance macro21(room4,door47,room7,boxB,door67,room6) is favoured by the chunk shown above to form the first part of a solution, resulting in a filtering out of any other undersirable instantiations. Note that this chunk suggests the initial position of the robot is irrelevant.

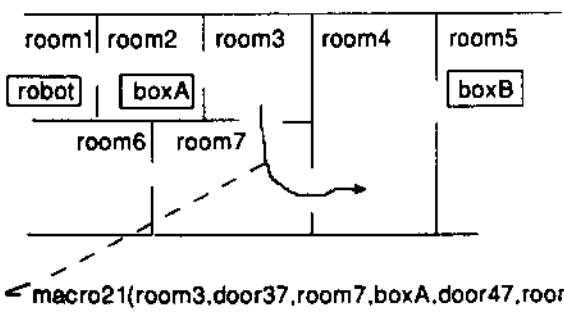

figure 1. (Note: doorXY connects roomX and roomY)

VI CONCLUSIONS

We have described a goal directed search which allows the use of weak methods for learning. Given a particular domain, these weak methods create strong heuristics, in the form of macros and b-chunks, through the experience of successful problem solving. The chunks record for each operator and generalised goal pair, the adviseable instantiations for operator variables. They do this by storing important similarities among the environment, initial state and goal in a form usable for future goal directed search. The number of possible instantiations of macros in the backward search tends to be much higher than primitives, and so the need for this heuristic pruning is greater.

We have used FM in several applications in which it builds up strong domain dependent heuristics by experience. Of particular note is the b-chunks' high degree of accross-task transfer of learning. This is because they record quite general similarities between the components of a problem space such that when these similarities are encountered again the choice of (macro) operator instantiation can be determined.

REFERENCES

1. Dawson, C. and Siklossy, L. "The Role of Preprocessing in Problem Solving Systems". In Proc. IJCAI-77, (1977).
2. Fikes, R., Hart, P. and Nilsson, N. "Learning and Executing Generalised Robot Plans", Artificial Intelligence 3, 251-288, (1972).
4. Iba, G. "Learning by Discovering Macros in Puzzle Solving". In Proc. IJCAI-85, (1985).
5. Kodratoff, I., "Careful Generalisation for Concept Learning", In Proc. ECAI-84, (1984).
6. Korf, R. E., "Macro Operators: A Weak Method for Learning", Artificial Intelligence 26, 35-77, (1985).
7. Laird, J., Rosenbloom, P. and Newell, A. "Chunking in Soar: The Anatomy of a General Learning Mechanism" Machine Learning 1, 11-46, (1986).
8. Laird, J., Rosenbloom, P. and Newell, A. "Toward Chunking as a General Learning Mechanism", In Proc. AAAI-84, (1984).
9. Langley, P. "Learning to Search: From Weak Methods to Domain-Specific Heuristics", Cognitive Science 9, 217-260, (1985).
10. Minton, S. "Selectively Generalising Plans for Problem Solving", In Proc. IJCAI-85 (1985).
11. Mitchell, T.M., Utgoff, P.E. and Banerji, R.B. "Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics", in "Machine Learning", Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), Tioga Publishing, (1983).
12. Mitchell, T.M. "Explanation-Based Genralisation: A Unifying View" Machine Learning 1, 47-80, (1986).
13. Porter, B. and Kibler, D. "Learning Operator Transformations", In Proc. AAAI, (1984).
14. Vere, S.A., "Induction of Relational Productions in the Presence of Background Information", In Proc. IJCAI-77 (1977).