Learning from Opportunities: Storing and Re-using Execution-Time Optimizations*

Kristian Hammond, Tim Converse and Mitchell Marks
Department of Computer Science
University of Chicago
1100 East 58th Street
Chicago, IL 60637

Abstract

In earlier work (Hammond 1986), we proposed a mechanism for learning from execution-time plan failure. In this paper, we suggest a corollary notion of learning from execution-time planning opportunities. We argue that both are special cases of learning from expectation failure (Schank 1982). The result of this type of learning is a set of plans for frequently occurring conjuncts of goals, indexed by the features in the world that predict their usefulness. We discuss this notion, using examples from the University of Chicago planner TRUCKER, an implementation of case-based planning in the domain of a UPS-like pick-up and delivery service.

1 Planning for conjunctive goals.

Current research in planning has taken a rather dramatic change in course. Fundamental to this change has been the acknowledgement that a planner cannot exhaustively pre-plan for a set of goals prior to execution. In part this change is the result of demonstrations that pre-planning for conjunctive goals is undecidable (Chapman 1987). A more important factor has been the realization that the dual assumptions of a closed world and complete knowledge of operators are untenable in any complex domain.

Alternative approaches to planning have included work in situated activity (Agre & Chapman 1987), reactive planning (Firby 1987), planning by analogy (Carbonell 1983) as well as approaches that include opportunistic elements (Birnbaum 1987). Although different, all of these approaches have included the use of domain-tailored methods for solving planning problems. In particular, they have used what amounts to libraries of plans for conjunctive goals specific to their domains. Rather than applying weak methods to construct new plans, these planners make use of these domain level knowledge structures.

As a result, much of these planners' knowledge consists of specific plans for multiple goals, optimized over parameters such as execution time, resource consumption or likelihood of success. Such plans are specific to particular conjuncts of goals, and trade off generality for efficiency. This trade-off ends up being profitable if the planner is able to correctly anticipate which sets of goals will arise and use those plans that deal with them.

*This work was supported in part by the Office of Naval Research under contract number N00014-88-K-0295.

This trend in planning suggests a need for work in learning: learning the particular plans to deal with conjunctive goal situations, and learning the features that predict their usefulness. In particular, it suggests that execution-time opportunities can be used to indicate when a planner should learn, and what it should learn. Along with this, however, a planner must be able to learn the features of the situation that actually predict the usefulness of the plan it has constructed. Using this type of execution-time information in learning is the topic of this paper.

2 Learning and plan interactions.

A great deal of work has been done in planning on the problem of interactions between the steps of plans (Sacerdoti 1975, Tate 1980, Dean, Miller & Firby 1986). Most of this work has been concerned with using temporal projection to tease out interactions at planning time. Unfortunately, this approach relies on the assumption that a planner can project all possible effects of its actions as well as anticipate all effects of the actions of other agents. In recent years, this assumption has essentially been abandoned by the planning community for all but the sparsest domains.

Sussman's HACKER (1975) was a clear counterexample to this approach in that it dealt with the problem of step interaction at *execution-time*. In this work, Sussman suggested that partial plans could be constructed and run in a *careful* mode during which a planner could observe and learn from the effects of the steps being run. Sussman's idea was that a planner could learn about negative interactions (*e.g.*, the effects of one step negating the preconditions required by a later one) and later on anticipate and thus avoid them.

This idea was expanded on and implemented in our own work in CHEF (Hammond 1986), in which we suggested that a planner could learn from its own planning failures. In particular we argued that a planner must be able to learn the conditions under which planning problems could later be predicted.

In CHEF, the focus was on learning about negative interactions between steps. In this paper, we explore how a planner might also learn from the positive interactions that it encounters during plan execution.

The difference between this approach and the one taken in CHEF lies in the relationship between expectations and plans. In CHEF, we studied *expectation failures* (Schank 1982) that corresponded to actual *plan failures*. In our current research, we are looking at expectation failures that are failures to anticipate *planning opportunities*. In CHEF, we argued that a planner has to respond to failure by repairing its current plan and by repairing the knowledge base (which is to say its expectations as to the results of its actions) which allowed it to create the plan. In this work, we argue that execution-time opportunities have to be responded to in a similar way: the planner should exploit the current opportunity *and* change its expectations so as to properly anticipate and exploit the opportunity in the future.

3 TRUCKER: Learning Plans for Conjunctive Goals.

TRUCKER is a project at the University of Chicago that explores issues of case-based planning and learning. The motivation behind the TRUCKER experiment is to study the learning of plans for conjunctive goals that tend to arise repeatedly in any domain.

TRUCKER's task and domain are simple to describe. It runs the operations of a multi-agent pick-up and delivery messenger service. It controls a fleet of trucks which roam a simulated city or neighborhood, picking up and dropping off parcels at designated addresses. Transport orders are "phoned in" by customers at various times during the simulated business day, and TRUCKER must see to it that all deliveries are successfully completed by their deadlines. This involves receiving requests from customers, deciding which truck should handle a given request, determining the order in which given parcels should be picked up and dropped off, figuring out routes for the trucks to follow, and monitoring the execution of the plans that have been constructed. A number of limited resources must be managed, including the trucks themselves, their gas and cargo space, and the planner's own planning and execution time.

TRUCKER starts off with very little information about the world that its trucks will be negotiating; all it has is the equivalent of a street map, an incomplete and potentially inaccurate schematic of its simulated world.

Conventional approaches to planning, with emphasis on exhaustive pre-planning, are inadequate to this task for a number of reasons:

- TRUCKER lacks complete information about its world.
- TRUCKER does not know all of its goals in advance – new calls come in that must be integrated with currently running plans.
- Planning time is limited. TRUCKER's world does not wait for it to complete plans before new events occur.
- Even given perfect advance information, an optimal solution to the problem TRUCKER faces is computationally intractable.

These constraints are not unique to the TRUCKER domain. They are constraints that have to be taken seriously in any complex domain.

Since pre-planning is impossible, and since at any point the system will have numerous goals that it is not able to satisfy at the moment, TRUCKER must plan *opportunistically*,[1] recognizing and acting upon opportunities for goal satisfaction as they arise. Much of TRUCKER's learning involves creating and storing plans that exploit these opportunities. TRUCKER also learns patterns of opportunity that it uses to recognize those situations for which its new plans are appropriate.

TRUCKER avoids the hard work of conjunctive goal planning unless execution-time cues indicate that such planning will be fruitful. Even with this restriction, however, the cost of such planning is high. Thus TRUCKER attempts to save the plans for conjunctive goals that it creates at execution time — with the idea that they can be applied again if these goals arise again.

The research into TRUCKER has led us to the following conclusions:

1. Optimal plans for conjunctive goals are breathtakingly hard to build.
2. Because they are hard to build, it is useful to cache plans for commonly occurring conjuncts of goals for later use.
3. The utility of these plans is undercut by their misapplication or missuggestion.
4. Because of this, it is important to reason not only about the content of a plan but also about its indexing in a plan library.
5. This indexing requires consideration of the likelihood of the same conjunct of goals appearing again.

The content of this paper is concerned with the last two points, which concern the reasoning required to index plans for conjunctive goals in memory. Before exploring the details of TRUCKER, however, it makes sense to look at a more commonplace example of the sort of planning knowledge that we are concerned with as well as the learning needed to acquire it.

4 Plans for conjunctive goals.

We can start with a simple example of the goal to have a quart of milk.

The basic plan for this goal is simple: go to the store, find the milk, buy it, and return home. During the execution of this plan a planner will have to move through the store looking for the milk. As he does so, he may see a bottle of orange juice and recall that he was out of it this morning. He also may recall that he was out of aluminum foil as well. How he notices these facts is not central here.[2]

At this point he does what any optimizing planner should do: he merges the separate plans for obtaining milk, orange juice and aluminum foil into a single plan for the conjunct of goals. He buys them all at once while at the store rather than buying them one at a time and returning home with each.

Here the planner has three options: he can forget that this particular version of the GROCERY-STORE plan exists, he can save the entire plan that satisfies all three goals or he can reason about what parts of the plan should be retained and how to index the resulting plan in memory.

The first option seems wrong on the face of it. This is a useful optimization of three separate instances of the

[1]While our concern with opportunism is in the same spirit as Hayes-Roth & Hayes-Roth's (1979) work on opportunistic planning, our work is aimed at the exploitation of *execution-time* rather than *planning-time* opportunism.

[2]For discussion of this issue see (Hammond, 1988)

GROCERY-STORE plan and could easily be re-used if these goals arise again.[3] The second option is better, but ignores that fact that the goal to have aluminum foil is not going to be activated all that often, while the goals associated with milk and orange juice are going to come up quite often. A plan for the entire conjunct, then, is going to be of little use. What seems to make the most sense is the third option—decide which parts of the plan should be retained and where the resulting structure should be indexed.

We want a planner that will take this experience and use it to form a new plan to pick up orange juice when it is at the store getting milk—without also picking up aluminum foil each time as well. The rationale for this choice of items to be included in this plan is clear. Given the rate of use of orange juice and milk, there is a good chance that at any given moment you may be out of either. Given the rate of use of aluminum foil, however, there is little chance that at any one time you will be out of it.

At this point, it is not enough to create the plan and associate it with the conjunctive goals of wanting to obtain milk and juice. This would give the planner the plan to pick up both in one trip but would not give it a plan that assumes that a set of goals is active in the presence of any one of them. What we really want is to have a plan that is activated when *any one* of the goals arises, and then checks for the existence of the other goals. That is, a plan that includes knowledge of the other plans with which it is typically merged.

To do this the planner must face a two-fold task. It must evaluate the likelihood that a similar conjunction will ever arise again - *i.e.*, determine if the plan is worth saving at all and which goals in the initial conjunct should be included. Then it must determine the set of features that predicts the presence of the conjunct. In the language of case-based planning, it must determine how to index the plan in memory.

We can approach this either empirically or analytically. The task can be done empirically, by trying the new plan when any one of the goals arises and removing links between it and those goals that do not predict the presence of the other goals. This is, in essence, the method implemented in the program IPP (Lebowitz, 1980). Or it can be done analytically, using explanation-based learning methods to construct explanations for why the goals should or should not be expected to arise in concert.

5 TRUCKER

Let's move on to a simple example from the TRUCKER program.

TRUCKER's task is to schedule pick-up and delivery requests. The conjuncts of goals that it looks for are conjuncts of such requests that can be fruitfully combined.

```
HANCOCK---<truck2>-------------------SEARS-TOWER
             \
              \
               \
             WATER---------WRIGLEY
```

Figure 1: Noticing an opportunity while driving.

For example, if TRUCKER is driving from the Hancock Building (HANCOCK) to Sears Tower (SEARS) in order to make a delivery, it notices that the Water Tower mall (WATER) is a pick-up point for another request (Water Tower to Wrigley Building (WRIGLEY)) that it has scheduled for a later time (figure 1). When it notices opportunities such as this TRUCKER has two problems to solve: first it has to find a useful merger of the two routes and then it must determine if the new plan will be worth saving.

6 TRUCKER's performance.

The central inputs to the planner are "phone calls" which it receives from a simulated world at various times in the day. These are delivery requests, which specify the pick-up address, the delivery address, and incidental information about the package such as size and contents.

TRUCKER's output is the actual execution of the plans it constructs for the trucks to follow to satisfy these requests. These plans are expressed in two levels of description: agendas and routes. The high-level agenda for a truck is a sequence of instructions about where to travel and what to do there. Typically it consists at any one time of alternating instructions for travel-steps and parcel transactions:

```
(GOTO (5802 S-WOODLAWN))
(PICKUP PARCEL-3)
(GOTO (920 E-55TH))
(DROPOFF PARCEL-5)
```

Routes are the plans that the trucks use to execute specific GOTO instructions. These are in the form of a list of the turns that have to be made while negotiating the route, described in terms of a street name and a compass direction. So the expansion of `(GOTO (920 E-55th))` after the pick-up is:

```
(START NORTH (5802 S-WOODLAWN))
(TURN EAST E-57TH)
(TURN NORTH S-CORNELL)
(TURN EAST E-55TH)
(STOP (900 E-55TH))
```

In order to execute a route, a truck (and thus TRUCKER) must parse the world it moves through, identifying landmarks, turns and drop off points. Trucks, then, are relatively "smart" executors in the world simulation, making this high level of plan specification sufficient for completion of deliveries.

7 Opportunities and Conjuncts

When TRUCKER is handed a new request, it adds the request to the agenda of one of its TRUCKS. As each

[3]One could argue that our planner *doesn't* need to learn the new plan in this example, in that he can just rebuild this plan from scratch the next time this situation arises. But this is just begging the question. It is easy to change the example slightly—by raising the cost of execution-time planning or obscuring the visual cues—thus making the anticipation of the goal conjunct far more valuable.

TRUCK moves through its agenda, it expands each GOTO instructions into a route, either by finding the route in its memory of past routes traveled or by constructing a new route from scratch using its map of the world.

TRUCKER merges requests in an effort to optimize over travel time. But it does so only when it encounters an opportunity to satisfy one request while it is actually running the route for a previous one. Initially, TRUCKER runs requests in order of "call-in". It also links each new request with the nodes in its representation associated with the locations that would serve as opportunities for satisfying the request. As the planner executes each stage of its plan, recognizing locations, it sometimes find requests associated with locations that it is passing. When this happens, TRUCKER considers the possibility that the new request could be merged with the current plan — as well as the possibility that the resulting route should be stored and re-used.

This is what happens in our example. In traversing a route from HANCOCK to SEARS tower, TRUCKER passes (and parses)the pickup point for the unsatisfied request WATER to WRIGLEY. Because its internal representation of the location WATER has been linked to the unsatisfied request, TRUCKER is reminded of the request as a result of recognizing location WATER. Once the possible connection is discovered TRUCKER attempts to find an optimal route that deals with both requests. Once this is done, it then saves the resulting route so that it can be re-used when the two goals arise again.

7.1 Placement of new requests in memory.

When TRUCKER receives a new request, it adds it to its current list of active requests, and in addition attempts to index it in memory under its representation of the locations of the pick-up and drop-off points. If that particular request has not been seen before, then a long-term record of the request is stored at those locations. If a request with the same pick-up and drop-off locations has previously been encountered, then the long-term record is updated with information about the new request. The long-term record also points to any currently active instantiations of itself, so that unsatisfied requests will be brought to the attention of the planner if their pick-up or drop-off points are encountered in the world.

In our example, when the request for a WATER pick-up and WRIGLEY delivery comes in, TRUCKER's knowledge of the two locations is marked by the fact that a request for a pick up and delivery between the two now exists. The request is also added to the queue of requests that TRUCKER determines its next action (Figure 2).

7.2 Detection of Opportunities.

As a truck negotiates a route, it is continually parsing its simulated world — in service of identifying turns and stops. As it does this, it activates the nodes in its place memory that represent the locations it passes. This in turn activates any requests that are associated with that location. The trucks are, in essence, looking at addresses, intersecting streets and visually distinctive landmarks and searching for requests for activity associated with those

```
HANCOCK-+---------------> REQ HANCOCK/SEARS
        |                 REQ UofC/UofI
        |  WATER----+--> REQ WATER/WRIGLEY
        |           |     REQ STANDARD/MAGML
        |           |
        |  WRIGLEY--+
SEARS---+

 Place Memory                  Request Queue
```

Figure 2: Requests in place memory and on the planning queue.

locations. If a pending request is discovered, this means that a possible opportunity for route combination has been encountered, and the planner is notified.

In our example, this is what happens as the truck passes location WATER. It activates the place memory node associated with the location, which in turn activates the suspended request for a WATER to WRIGLEY delivery. At this point TRUCKER is notified that an opportunity to merge two requests has been detected.

7.3 Capitalizing on Opportunities.

When the planner is alerted to the fact that a truck is passing by the pick-up point of an active request, it considers the possibility that it can merge its current plan with the newly activated request. This involves three steps. First TRUCKER inspects the truck's agenda to see what request the truck is currently engaged in fulfilling, and what locations the truck will be visiting next. Next, it consults the map and exhaustively examines the different orderings of the locations in the newly activated request and those in its current request agenda. The goal here is to find the shortest ordering that will combine the two requests. It then applies a simple geometrical heuristic to the ordering produced to determine whether the combination is "reasonable" (the points may well be configured in such a way that even the optimal combination is not particularly efficient). If a good combination is found that can be executed by the given truck, then the truck is directed to pick up the parcel, and its agenda is updated appropriately.

Determination of the optimal combination involves a considerable amount of computation. It is important to note that this computation is only performed when the planner has a good reason to believe that a particular combination might be fruitful. At no point does the planner attempt to exhaustively test different combinations of the routes it knows about.

In our example, TRUCKER is able to construct a route that takes it from HANCOCK to SEARS by way of WATER and WRIGLEY that adds very little distance to the initial route. With the best route requiring over significantly less travel to accomplish WATER to WRIGLEY than running the routes independently, TRUCKER allows the optimization to stand.

7.4 Storage of new plans.

Whenever the planner is alerted to the possibility of combining two requests, some annotation is made to the long-term record of those requests. The content of this annotation depends on the result of the attempt to combine the two. If a good combination is found, both requests have a notation added to them that points to the other request of the pair and includes the combination. This means that the computation involved in producing the combination need never be repeated. If no fruitful combination is found, the annotation includes that information in place of the combination. This is to prevent a "rediscovery" of the opportunity and subsequent fruitless search for a combination.

In the above example, the TRUCKER's knowledge of HANCOCK/SEARS requests is annotated with the information that they can be fruitfully combined with WATER/WRIGLEY requests. The same sort of annotation is added to TRUCKER's knowledge of goals involving WATER/WRIGLEY deliveries. Each request points to its optimal route as well as routes that are associated with fruitful request conjunctions that TRUCKER has discovered.

7.5 Re-use of plans.

Because it is saving routes for combinations of requests, TRUCKER is able to replan for those combinations with little or no effort.

When the TRUCKER receives a request, the request's long-term record is inspected to see if there are any notations about combinations with other requests. If so, the planner looks to see if the other requests are currently active. If it finds the requests that it has previously been able to merge with the current one, the plan for the conjunction is added to the agenda rather than those for the individual requests. It is important to note that plans for combination of requests are only activated when all the requests are active.

In the above example, if the planner receives a new HANCOCK/SEARS request on another day, it would discover the notation about the WATER/WRIGLEY request. If no active WATER/WRIGLEY request was found, the planner would proceed as normal with HANCOCK/SEARS. If a WATER/WRIGLEY is found, however, the conjoined version of the two requests is added to the agenda.

Lacking a WATER/WRIGLEY request, the HANCOCK/SEARS request is just added to the agenda. If TRUCKER subsequently gets a WATER/WRIGLEY request, it will find a notation about HANCOCK/SEARS. Assuming that the HANCOCK/SEARS request has not already been satisfied, the planner will find HANCOCK/SEARS among its active requests, and substitute the stored plan for the conjunct (HANCOCK/WATER/WRIGLEY/SEARS) in place of the HANCOCK/SEARS plan without further computation.

8 Conclusions

TRUCKER uses domain knowledge and empirical validation to construct and index plans for conjunctive goals in a plan library. It constructs these plans when the opportunity to merge two goals suggests itself during execution, and indexes the resulting plan in a way that makes it possible to re-use the optimization when the opportunity to do so arises again. With the addition of domain knowledge concerning the the underlying causes of the requests that TRUCKER plans for, it is possible to further discover the exact circumstances under which the plans are applicable.

9 Acknowledgements

We would like to thank Gregg Collins for his comments and conversation on this paper. We also thank Tom McDougal and Jeff Berger for their critical comments, and Oleg Voloshin and Alex Beels for their help in the initial TRUCKER implementation.

10 References

Agre, P.E. and Chapman, D., Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*, AAAI, Seattle, WA, July 1987, 268-272.

Birnbaum, L., *Integrated Processing in Planning and Understanding*, Yale Technical Report # 480. (New Haven, CT, 1986).

Carbonell, J.G., Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning, An Artificial Intelligence Approach.* (Tioga, Palo Alto, CA, 1983) 137-162.

Chapman, D., *Planning for Conjunctive Goals*, Technical Report TR 802, MIT Artificial Intelligence Laboratory, 1985.

Dean, T., Firby, R. J., Miller, D., *The Forbin Paper*, Technical Report 550, Yale University Computer Science Department, July 1987.

Hammond, K., Learning to anticipate and avoid planning problems through the explanation of failures. In *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Philadelphia, PA, 1986.

Hammond, K. Opportunistic Memory: Storing and recalling suspended goals. In *Proceedings: Case-based Reasoning Workshop* (Clearwater Beach, FL) May 1988, 154 - 169.

Hayes-Roth, B., and Hayes-Roth, F., A cognitive model of planning. In *Cognitive Science*, 2, 1979, 275-310.

Lebowitz, M., *Generalization and Memory in an Integrated Understanding System*, Ph.D. Thesis, Yale University, October 1980.

Sacerdoti, E.D., *A structure for plans and behavior*, Technical Report 109, SRI Artificial Intelligence Center, 1975.

Schank, R. *Dynamic Memory: A theory of reminding and learning in computers and people.* Cambridge University Press, 1982

Sussman, G.J., *HACKER: a computational model of skill acquisition*, Memorandum 297, MIT Artificial Intelligence Laboratory, 1973.

Tate, A., *INTERPLAN: A plan generation system which can deal with interactions between goals*, Research Memorandum MIP-R-809, Machine Intelligence Research Unit, University of Edinburgh, 1974.