Pluto: Managing Multistrategy Learning Through Planning

Gordon T. Shippey J. William Murdock Ashwin Ram

Georgia Institute of Technology
College of Computing
Atlanta, Georgia 30322-0280
(shippey|murdock|ashwin)@cc.gatech.edu

Multistrategy learning systems are systems that employ multiple methods to solve learning problems. In many multistrategy systems, either the user or the system selects a single method to use on the current problem. At best, such a selection-type multistrategy learning system can solve the union of the problems solvable by the individual learning methods it contains. Another strategy is to have the user specify a sequence of methods to apply to a particular problem at compile time. Both of these strategies fails to tap the full potential of the learning methods under their control.

One way to overcome this limitation is to cast the learning task as a planning problem. By treating learning strategies as operators in a planning problem, several learning strategies can be linked together to form a network bridging the gap between the system's initial knowledge state, current inputs, and some knowledge goal. Coordinated networks of learning strategies can solve problems beyond the range of any individual learning strategy.

We are currently developing a computational model of the planning-to-learn process which we call PLUTO (Planning to Learn Using Transmutation Operators). The current version of PLUTO creates and executes learning plans in the domain of consumer decision making.

```
(defframe '(cheap-low-emis-vacuums
    (isa set)
    (members (?q
           (var-type knowledge-goal)))
    (restrictions (cheap-restriction
              (isa constraint)
              (constraint-type fuzzy-constraint)
              (slot-name :relation price)
              (goal minimize)
              (reference vacuum-cleaner))
           (low-emissions-restriction
              (isa constraint)
              (constraint-type fuzzy-constraint)
              (slot-name :relation emissions)
              (goal maximize)
              (reference vacuum-cleaner)))))
```

Figure 1. A knowledge goal in the PLUTO system.

PLUTO describes the knowledge it wishes to learn in the form of knowledge goals like the one shown in figure 1. PLUTO's knowledge goals are frames with variables holding the place of desired information. The knowledge goal in figure 1 could be interpreted as "find the set of vacuum cleaners which are inexpensive and low in emissions (i.e. vacuums that don't let dust inside the vacuum escape)". Note that this knowledge goal does not specify a specific price or a specific emissions level desired. The variable `?q` is holding the place for the members of this set frame.

Given the knowledge goal in figure 1, a list of vacuum cleaner models with prices and emissions ratings, plus a library of knowledge operators, PLUTO can create several different plans to learn which of the vacuum cleaners in the knowledge base satisfy the requirements of the knowledge goal. Plans are generated by matching operators to open knowledge goals. Matching an operator to a knowledge goal produces an operator instance which may require new open knowledge goals to perform its task. For instance, an operator may break the goal to find vacuums that are cheap and low in emissions into two independent goals: find vacuums that are cheap, and find vacuums that are low in emissions. When a plan has no open knowledge goals, it is considered to be a finished plan and ready for execution.

During execution each operator instance in the plan takes its input from either the knowledge base or the output of other operators and then generates its own results, which will either be part of the final solution, or input for later operators to use as input.

There are several noteworthy properties of the planning and execution processes. First, PLUTO autonomously creates and executes learning plans. Second, PLUTO uses multiple knowledge operators in its plans. While none of the operators alone are sufficient to solve the problems, PLUTO can combine them into viable solutions, extending PLUTO's range beyond that of the select-and-apply multistrategy learning methods. Third, different instantiations of the same knowledge operator can be used in different contexts, even within the same plan. Fourth, PLUTO can generate several different plans to solve the same problem. Since plans sometimes fail, multiple plans give PLUTO multiple chances to succeed. For instance, there are multiple methods available to PLUTO to define "cheap". Fifth, a single plan may be valid over a space of knowledge goals, which makes plans a reusable resource for learning.