Learning to Improve Quality of the Plans Produced by Partial Order Planners

Muhammad Afzal Upal **Reneé Elio**

Department of Computing Science
University of Alberta, Edmonton
Canada, T6G 2H1

Considerable planning and learning research has been devoted to the problem of automatically acquiring search control knowledge to improve planning efficiency. However, most speed up learning systems define planning success rather narrowly, namely as the production of any plan that satisfies the goals regardless of the quality of the plan. As planning systems are applied to real world problems, concern for plan quality becomes crucial. Many researchers have pointed out that generating good quality plans is essential if planning systems are to be applied to practical problems (Wilkins 1988; Drabble, Gil, & Tate 1995; Perez 1996; Ruby & Kibler 1993).

The problem is that in most practical situations we only have the *post-facto*[1] knowledge about plan quality. Such knowledge allows us to determine the quality of a plan once complete plans have been generated but it is of little use during plan construction. The learning problem then is that of translating the *post-facto* quality measurement knowledge into *operational* knowledge that can be used during plan generation to guide the planner towards making choices that lead to better plans. To address this issue, we outline a technique of learning to improve the quality of plans generated by partial-order planners. This technique can be integrated with speed-up learning methods (such as (Ihrig & Kambhampati 1997)'s DerSNLP). We call the resulting approach a Performance Improving Partial-order Planner (PIPP). This paper presents the key ideas underlying this approach.

The *post-facto* Plan Quality Measurement Knowledge

Consider the following observations:

> Real-world problems *are* multi-objective... Each of our lives is filled daily with multiobjective problems. Should I take the car or the bus? Well, the bus is cheaper (when the cost of gasoline, maintenance and insurance are computed for the car), but the car is more convenient, particularly since I should stop at the store on my way home from work. The bus will save energy, but I can listen to the radio in the car. There are probably other attributes or objectives in addition to cost, convenience, energy consumption and comfort that might be considered in choosing between the car and the bus... Problems like this abound. (Cohon 1978)

We call the plan quality measurement knowledge to be *complex* if it depends on multiple *variant* quality factors (or metrics) and *simple* if it depends on a single *static* quality metric. A static quality metric assigns a fixed value to each action whereas a variant quality metrics may assign different values to the same action depending upon the situation in which that action is applied. For instance, a simple quality measure would be $\frac{1}{plan\ length}$. The amount of fuel consumed in traveling between various locations is an example of the variant quality metric as the amount of fuel varies depending on the distances between the locations. A few learning systems that do possess quality knowledge (Perez 1996; Iwamoto 1994), all assume simple quality knowledge.

Unfortunately, in most practical domains, operators need to be expressive and general enough to be applicable in a number of situations (such as $drive(X, Y)$ to denote the action of driving from a location X to a location Y) and the value of a quality metric (such as fuel-cost) cannot stay the same for an action in all the situations. Also, as a number of planning and operations research experts have noted, in most real world scenarios plan quality depends on a number of competing factors. We agree with Keeney and Raiffa (Keeney & Raiffa 1993) that most interesting planning problems are multiobjective.

Value-theoretic functions are a well-developed mechanism devised by operation research workers to represent the evaluation function for multiobjective problems. A value function is defined on the outcome (i.e., the final-state) of a complete plan. The world states are described by a finite number of attributes. The first task towards the formulation of a value function is identification of the decision attributes. Keeney and Raiffa (Keeney & Raiffa 1993) suggest a hierarchical refinement scheme starting with the *highest level objec-*

[1]The term *post-facto* was first defined by Perez (Perez 1996fi to refer to the quality knowledge that can only be used to measure plan quality only *after* planning.

tives and refining them down to the *low level measurable* attributes. For instance, the overall objective of an agent using a transportation system maybe to "have a good trip" which can be refined down to the measurable attributes such as "minimize door-to-door travel time" and "minimize fare costs." Once various objectives have been identified, the next step is to elicit the degree of user's degree of preference of one attribute over another. Operations research and choice modeling researchers study different techniques for eliciting domain expert's preference knowledge (Hensher 1981; de Soete & Hubert 1990). Based on the expert's responses to various combinations of multiple decision attributes, techniques such as conjoint analysis (Louviere 1988) are used to estimate attribute utilities and to encode the revealed preference structure into a value function V.

$$V : D \times D \rightarrow \Re$$

where D is the set of decision attributes and $\Re$ is the set of real numbers.

If an agent's preferences constitute a partial-order over outcomes and satisfy certain rationality criteria (such as transitivity), the central theorem of decision theory (Fishburn 1970) states that these preferences can be represented by a real-valued *value function* V such that if s_1 and s_2 denote two outcomes then s_1 is preferable to s_2 i.e., $s_1 \succ s_2$ iff $V(s_1) > V(s_2)$. Even if the agent's preferences do not form a partial-order, the value function can still be used to form good approximations (Yu 1985). We believe that value functions are expressive enough to denote the *post-facto* plan quality measurement knowledge. And indeed, many AI planning researchers (Williamson 1996; Ephrati, Pollack, & Milshtein 1996; Haddawy & Hanks 1998) have used value functions to solve AI planning and reasoning tasks.

The Operational Quality Knowledge

Definition 1 Operational knowledge *o is a function from the set of partial plans to the set of refinement sequences.*

Operational knowledge can be considered to be a set of operational rules.

Definition 2 An operational rule *has a function s defined on a partial plan as its antecedent and planning decision sequence as its consequent.*

$$s(PP_m) \rightarrow d_i, d_{i+1}, ...d_j \qquad (1)$$

s is a Boolean function (such as the viability function *which is true if the refinement sequence $d_i, ..., d_j$ is applicable to the partial plan PP_m) defined on the partial plans.*

We define $(j - i)$ to be the *specificity* of an operational rule. Specificity of a sequence of planning decisions that, when applied to an initial plan, refines it into a flawless plan is defined to be infinite. We refer to the operational rules of infinite specificity as *global.* The term *local rule* refers to an operational rule of zero specificity. For instance, a rejection rule learned by SNLP+EBL (Kambhampati, Katukam, & Qu 1996) can be considered to be a local rule. A DerSNLP-case can be considered to be a global rule because it evaluates the initial partial plan PP_1, obtained by adding the dummy actions to the null plan, to see if its effect set E has the *relevant initial conditions* and provides the complete planning trace if it does. Relevant initial conditions are the conditions that are needed as preconditions for actions in the final plan. Thus function s, in this case, is defined to be a function that simply evaluates the relevant initial conditions $s = (evaluate\ (init\text{-}conds\ PP_1))$. Presence of these conditions in the new problem specification guarantees that the plan created under the guidance of the retrieved case will satisfy the currently active goals but not that it will lead to a high quality plan.

Definition 3 An operational quality rule *has as its antecedent a function s' obtained by adding a* betterness-conditions *function b to s. The consequent specifies that a decision sequence D_x is preferable to the decisions sequences $D_{x+1}, ..., D_y$.*

$$s'(PP_m) \rightarrow\ prefer\ D_x\ over\ D_{x+1}, ..., D_y \qquad (2)$$

where $s'(PP_m) = s(PP_m) + b(PP_m)$, $b(PP_m)$ is true when applying decision sequence D_x to the partial plan PP_m leads to a better quality solution than the decision sequences $D_{x+1}, ..., D_y$ and false otherwise.

We'll define $y - x$ as the *size* of an operational quality rule.

Definition 4 Operational quality knowledge *is a collection of the operational quality rules.*

An operational rule is an operational quality rule of size zero and with its betterness-condition function set to true. Thus operational knowledge is a special case of operational quality knowledge.

In the next section, we describe the learning and planning framework, PIPP, that can acquire operational quality knowledge from its own planning experiences or from the plans provided by a user. Given a problem description and a plan that solves that problem, PIPP creates a trace of the given plan and by comparing it with the trace of its own plan it learns operational quality rules.

PIPP

Knowledge Representation

Most planners support a version of the STRIPS language to represent states and actions. In STRIPS, states are represented by conjunctions of propositional-attributes (represented by function-free ground literals) and actions are represented by three components:

- *The action description:* The parametrized name for a possible action such as $MOVE(O, X, Y)$ denotes the action of moving O from X to Y.

- *The Preconditions:* A conjunction of propositional attributes that must hold for the action to work.
- *The effects:* A conjunction of propositional attributes that describes as to how the world changes by the application of the action. The effects are described by *add* and *delete* lists of propositional attributes made true and false (respectively) by the execution of the action.

Various extensions have been developed by practical planning researchers (Tate, Drabble, & Kirby 1994; Wilkins 1988; Williamson 1996) to support metric attributes that are needed to denote numerical quantities of resources. Metric attributes are essentially treated like propositional attributes in the way they enter the state description and an action's preconditions and effects. The main difference is that while propositional attributes are logical conjunctions, metric attributes also involve numerical expressions. PIPP uses R-STRIPS, an extension of STRIPS suggested by (Williamson 1996).

In R-STRIPS, the world states are described in terms of attributes which may be *propositional* or *metric*. The following description of R-STRIPS is adapted from (Williamson 1996)[2]:

Definition 5 (State): *An R-STRIPS state is a 2-tuple* $< S_p, S_m >$ *where* S_p *denotes propositional attributes and* S_m *denotes metric attributes.*

Definition 6 (Propositional Attribute): *A propositional attribute is a 2-tuple* $< n, v >$ *where* n *is the symbol denoting proposition name and* v *is the proposition value.*

Definition 7 (Metric Attribute): *A metric attribute is a formula* $< \beta, l >$ *where* β *is a symbol denoting a resource name and* l *is a real number denoting the amount or level of that resource.*

Definition 8 (Metric Precondition): *The metric precondition of an action* α *is a formula* $< \beta, F_{\alpha\beta} >$ *where* β *is a resource and* $F_{\alpha\beta}$ *is a metric precondition function defined over all possible bindings of* α*'s parameters* $< p_1, \ldots, p_n >$. $F_{\alpha\beta}$ *is a boolean function.*

Definition 9 (Metric Effect): *The metric effect of an action* α *is a formula* $< \beta, F_{\alpha\beta} >$ *where* β *is a resource and* $F_{\alpha\beta}$ *is a metric effect function defined over all possible bindings of* α*'s parameters* $< p_1, \ldots, p_n >$.

Definition 10 (R-STRIPS Action Schema): *An R-STRIPS action schema is a sex-tuple* $\alpha =< \alpha_n, \alpha_v, \alpha_{pp}, \alpha_{mp}, \alpha_{pe}, \alpha_{me} >$ *where*

- α_n *denotes the symbolic name,*
- α_v *is a list of variable parameters,*
- α_{pp} *denotes propositional preconditions.*
- α_{mp} *is a set of metric preconditions,*
- α_{pe} *denotes propositional effects, and*
- $\alpha_{me} = < \beta, F_{a\beta} > |$ *for each resource* β *is a set of metric effects.*

Definition 11 (Ground Action): *A ground action is an action-schema in which all variables have been bound to object symbols in the domain.*

A ground action represents a mapping between world states. This mapping is defined over those states in which the action is viable.

Definition 12 (Viability of an action): *An action* α *is viable in a state* $S =< S_p, S_m >$ *if* $\alpha_{pp} \subset S_p$

$$\forall(< \beta, F_{\alpha\beta} > \in \alpha_{mp}),\ F_{\alpha\beta}(S) \text{ is true.}$$

Definition 13 (Action Execution): *The execution of a viable action* α *in a world state* $S =< S_p, S_m >$ *is a new world state* $S' =< S'_p, S'_m >$ *such that*

$$S'_p = \alpha_{pe} S_p$$

and

$$S'_m = \{< \beta, l + F_{\alpha\beta} > | < \beta, l > \in S_m\}$$

Definition 14 (Plan): *A plan is an ordered sequence of ground action schemas.*

Definition 15 (Plan Viability): *A plan* $p =< a_1, \ldots a_n >$ *is viable in a state* S_1 *if each action* a_i $1 \leq i \leq n$ *is viable in the state* S_i *where* $S_i = a_{i-1}(S_{i-1})$.

Definition 16 (Plan Outcome): *Outcome of a plan is the final-state achieved by executing the plan in the initial-state.*

Here, we assume that all the decision attributes can be specified as metric attributes i.e., the quality function only depends on the metric attributes (i.e., the resource levels) in the final state.

The PIPP Approach

Unlike conventional analytical learning systems (such as EBL systems that learn from one successful or unsuccessful solution) and like QUALITY (Perez 1996), PIPP performs intra-solution learning. Given two plans of possibly differing quality that achieve the same goal, the intra-solution learning process involves identifying the *conflicting choice points* and forming an explanation as to under what conditions in the initial state the sequence and (subsequence of) the better planning episode's refinement choices are better than the sequence (and subsequences of) the worse planning episode choices. A conflicting choice point is a decision point in the trace of the better plan where the refinement used by the better plan differs from the system's default choice. Each conflicting choice point indicates a gap in the system's control knowledge and hence a need to learn. The generalized explanation becomes the

[2] Williamson's original formulation of R-STRIPS also allowed for partially satisfiable goals. We've simplified R-STRIPS to reflect the fact PIPP does not reason with partially satisfiable goals. Williamson also defines the outcome of a plan to include the intermediate states as well as the final state.

betterness conditions because it identifies the most general conditions under which the better plan (and hence the decision sequence that led to it) is better than the worse plan (and the decisions sequence that led to it). We prove this assertion in the following theorem.

Theorem 1 *Given*

- *an initial-state* $I =< I_p, I_m >$ *where* $I_m = \{< i_1, l_{01} >, \ldots < i_k, l_{0k} >\}$
- *a plan* $p_1 = a_1, \ldots a_n$
- *a plan* $p_2 = b_1, \ldots b_m$
- *a quality function* q
- p_1*'s outcome (when executed in I)* $=< G_p, G_m >$ *where* $G_m = \{< i_1, l_{g_1 1} >, \ldots < i_k, l_{g_1 k} >\}$
- p_2*'s outcome (when executed in I)* $=< G'_p, G'_m >$ *where* $G'_m = \{< i_1, l_{g_2 1} >, \ldots < i_k, l_{g_2 k} >\}$.

Then

1. p_1 *and* p_2 *are viable in I iff the union of the relevant initial conditions of* p_1 *and* p_2 *is a subset of I.*
2. p_1 *is better than* p_2 *iff the betterness conditions of* p_1 *over* p_2 *are satisfied.*
3. *The betterness conditions can be computed in* $O(k * (n + m))$ *time.*

Proof:

1. Assertion 1 follows from the definitions of plan viability and the relevant initial conditions.
2. p_1 is better than p_2 iff p_1's outcome has a higher quality value than p_2's outcome i.e.,

$$q(l_{g_1 1}, \ldots l_{g_1 k}) > q(l_{g_2 1}, \ldots l_{g_2 k}) \quad (3)$$

   The value of a metric attribute in the final-state g is equal to the value of that attribute in the state $g - 1$ plus the value of the metric effect function for the final action. We can recursively carry this argument all the way back to the initial-state to get the following values of the metric attributes in the final state.

$$l_{g_1 i} = F_{a_n i} + F_{a_{n-1} i} + \ldots + F_{a_1 i} + l_{0i} \quad (4)$$

$$l_{g_2 i} = F_{b_m i} + F_{b_{m-1} i} + \ldots + F_{b_1 i} + l_{0i} \quad (5)$$

   Where F_{ij} is the metric effect function associated with the attribute j and action i. Substituting these values in inequality 3 transforms q into a function q' only of the metric attribute values in the initial-state. Let's denote the metric attribute values in the initial state by the variables $X_1 = l_{01}, \ldots X_k = l_{0k}$ then substituting these values in inequality 1 we get

$$q'(p_1, X_1, \ldots, X_k) > q'(p_2, X_1, \ldots, X_n) \quad (6)$$

   The Inequality 6 represents the betterness conditions of p_1 over p_2. Clearly, by formulation we can see that p_1 is better than p_2 iff 6 holds.
3. In order to compute the betterness conditions, the following steps have to be performed.
   (a) Assign symbolic values $X_1, \ldots, X_k$ to the the metric attribute levels in the initial-state.
   (b) For $i = 1$ to k (i.e., for each of the k attributes)
      i. For $j = 1$ to n do (i.e., for each action in p_1)
      A. $l_i(S_j) = l_i(S_{j-1}) + F_{a_j i}$ (i.e., compute the value l_i of metric attribute i in state S_j)
      i. For $j = 1$ to m do (i.e., for each action in p_2)
      A. $l_i(S_j) = l_i(S_{j-1}) + F_{b_j i}$ (i.e., compute the value l_i of metric attribute i in state S_j)
   (c) Compute the quality function q by doing a symbolic substitution of the values of metric attribute values computed in the steps (a) and (b).

   We know that the above symbolic computations can be performed because we can reasonably assume that the metric effect functions and the quality function can be encoded as closed formed expressions.
   Complexity: The step (b) dominates the whole function. Hence the algorithm is of $O(k * (m+n))$ complexity.

For each conflicting choice point, a learner can learn local and/or global operational quality rules and/or rules of any specificity between the extremes of local and global. The local rules suggest a preferred alternative at a particular decision point, but that decision may affect other decisions that the planner should make at other points. Indeed in some situations, making a choice at some decisions point d_i may only be preferable if another choice is made at decision point $d_j, i \neq j$. Global knowledge encodes preference for a number of choices at various decision points in a complete plan. The local rules, however, are more general than the specific global rules and are applicable in a broader number of situations. $PIPP_{LOCAL}$ is a module of PIPP that learns local rules and $PIPP_{GLOBAL}$ learns global rules.

$PIPP_{GLOBAL}$: The betterness conditions are conjunctively combined with the union of the relevant initial conditions of the system's plan and the external agent's plan learned by the derivational analogy module to form the antecedents of a global rule. The consequent of the global rule says "prefer the entire sequence of the better planning episode's decisions over the entire sequence of the worse planning episode's decisions."

$PIPP_{LOCAL}$: The betterness conditions for a conflicting choice point and the unconsumed relevant initial conditions form the antecedent of a local rule. Unconsumed relevant initial conditions are those relevant initial conditions that have not been consumed by any causal link added to the partial plan so far. The consequent of a the local rule says "prefer the refinement applied by the better plan over the the refinement applied by the worse plan at this decision point."

The operational rules learned in each learning episode are stored in the rule-memory and retrieved when a similar problem is presented to the system. If more than one rule is applicable to a partial plan, it prefers the rule that has the largest size and if they are of equal size the one that is the most specific.

For each conflicting choice point, PIPP needs to compute the complete plan following the system's default choice to compare it with the external agent's plan to derive the betterness conditions. How much effort does the system needs to put into learning depends on how different the external agent's choices are from the system's choices i.e., how much does the system need to learn? Initially, the system may need to expend more in learning but as it acquires more knowledge the number of conflicting choice points should decrease. At worst, for a plan-tree that is n-nodes deep and all the choice points are conflicting, we may end up computing n plans but in general, that should not be the case.

Architecture

As shown in figure 1, PIPP has three components: a case-based reasoner, a generative partial-order planner and an inference engine. The planning component generates a trace of the external agent's plan, identifies the conflicting choice points and the relevant initial conditions. The inference engine compares two plans for quality using the complex quality measurement knowledge provided, and generates betterness conditions from the trace of the comparison. The case-base stores and retrieves the learned operational quality rules. The antecedent of an operational quality rule is used as the indexing scheme to store and retrieve its consequent.

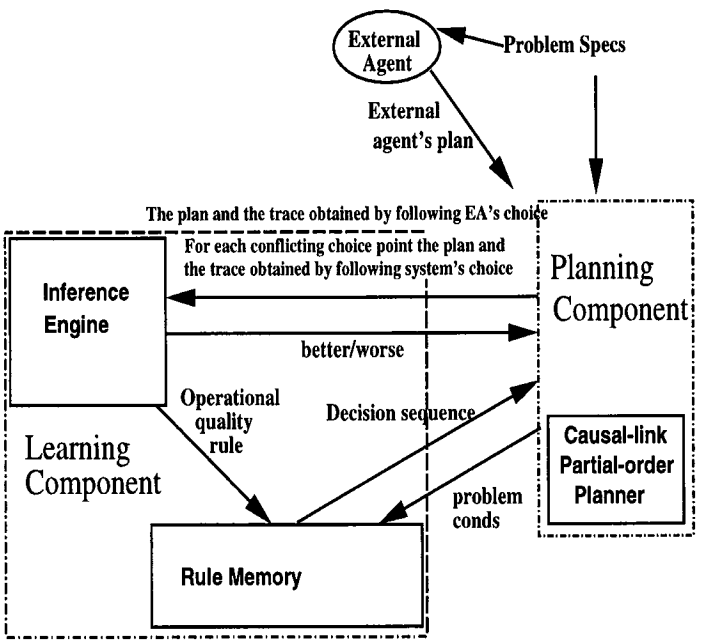

Figure 1: The PIPP Architecture

PIPP's Algorithm

- *Input:* Problem description in terms of the initial state I and the goal G. External agent's plan pe.

PIPP$(< I, G >, pe)$

1. generate-plan$(< I, G >, p, tr(p), rc(p))$
2. If better-plan(p, pe, bc) then
   (a) generate-trace$(< I, G >, pe, tr(pe), rc(pe)$, Plans, Conf-decs, Tr, RC, URC)
   (b) pipp-global$(bc, rc(p), rc(pe), tr(p), tr(pe))$
   (c) pipp-local$(bc, urc(p), urc(pe), d(p), d(pe))$
   (d) For each conflicting choice point $i > 1$,
      i. If better-plan(p_i, pe, bc_i) then
      A. pipp-global$(bc_i, rc(p_i), rc(pe), tr(p_i), tr(pe))$
      B. pipp-local$(bc_i, urc(p_i), urc(pe), d(p_i), d(pe_i))$

Generate-plan$(< I, G >, p, tr(p), rc(p))$, given a problem description $< I, G >$ generates a plan p and the planning trace $tr(p)$ and relevant initial conditions $rc(p)$ for this plan.
Generate-trace$(< I, G >, pe, tr(pe), rc(pe)$, *Plans, Conf-decs, Tr, RC, URC)* given a problem description $< I, G >$ and a plan pe generates

- the planning trace $tr(p)$
- relevant initial conditions $rc(pe)$ for this pe
- Set of n conflicting decisions Conf-decs $= \{< d(p_i), d(pe_i) > | 1 \leq i \geq n\}$.
- Set of n plans, Plans, produced by following system's default choice at each conflicting choice point Plans $= p_i | 1 \leq i \geq n\}$
- Set of n traces, Tr, for each plan in Plans Tr $= tr(p_i) | 1 \leq i \geq n\}$
- Set of n relevant initial conditions, RC, for each plan in Plans RC $= rc(p_i) | 1 \leq i \geq n\}$
- Set of n unconsumed relevant initial conditions, URC, for each plan in Plans Tr $= urc(p_i) | 1 \leq i \geq n\}$

The Boolean function *better-plan*(p, pe, bc) takes two plans, p and pe, and returns true if p has higher quality than pe and generates the betterness conditions, bc, of p over pe.
Pipp-global$(bc, rc(p), rc(pe), tr(p), tr(pe))$ forms and stores a global rule

$$bc + rc(p) + rc(pe) \rightarrow \text{prefer } tr(p) \text{ over} tr(pe).$$

Pipp-global$(bc, urc(p), urc(pe), d(p), d(pe))$ forms and stores a global rule

$$bc + urc(p) + urc(pe) \rightarrow \text{prefer } d(p) \text{ over} d(pe).$$

Domain and Some Examples

Modified Logistics Transportation Domain

In the transportation domain of Veloso (Veloso 1994), packages must be delivered to different locations in several cities. Trucks are used to transport packages within the same city, and planes are used to transport packages between different cities. We've modified the original transportation domain so that there is more

than one way of going from one city to another and there are some resources to reason about. The action *MOVE-TRUCK-ACITIES* was added as an alternative means of moving between the cities. States are described by propositional as well as metric attributes of money and time. The places (i.e., airports *AP* and ports *PO*) have *positions*. We modified the action descriptions so that metric preconditions of each action specify the amount of money required for the action and the metric effects specify how the action changes the amount of money and the time in the world. For instance, the time-taken and the cost of (*MOVE-TRUCK OBx Ply Plz*) is defined by a function of the weight of the object *OBx* to be transported as well as distance between *Ply* and *Plz*. Problems are produced by generating random initial states and goals. Package weights and place positions are also assigned random values. If places are in the *SAME-CITY* distances between them are generated to be less than a *short-distance*, where distance between the places *Plx* and *Ply* is calculated as

$$(distance\ Plx\ Ply) = |(position\ Plx) - (position\ Ply)|.$$

The plan quality function is defined on the metric attributes of *time* and *money*,

$$q(time, money) = 5 * time - money.$$

The actions schemas are stated as follows:

```
(action (load-truck  ?O ?PL ?AP)
 preconds ((b-at-object ?O ?AP)
  (c-at-truck ?PL ?AP) (> money 5))
 effects ((b-ainside-truck ?O ?PL)
  (not (b-at-object ?O ?AP))
  (money -5) (time 5)))

(action (unload-truck  ?O ?PL ?AP)
 preconds ((c-at-truck ?PL ?AP)
  (b-ainside-truck ?O ?PL))
 effects ((b-at-object ?O ?AP)
  (not (b-ainside-truck ?O ?PL))
  (money -5) (time 5)))

(action (move-truck ?PL ?L ?M)
 preconds ((a-same-city ?L ?M)
  (c-at-truck ?PL ?L))
 effects ((c-at-truck ?PL ?M)
  (not (c-at-truck ?PL ?L))
  (money (* -10 (distance ?L ?M)))
  (time (* 10 (distance ?L ?M)))))

(action (load-plane  ?O ?PL ?AP)
 preconds ((b-at-object ?O ?AP)
  (c-at-plane ?PL ?AP))
 effects ((b-inside-plane ?O ?PL)
  (not (b-at-object ?O ?AP))
  (money -5) (time 15)))

(action (unload-plane  ?O ?PL ?AP)
 preconds ((c-at-plane ?PL ?AP)
  (b-inside-plane ?O ?PL))
 effects ((b-at-object ?O ?AP)
  (not (b-inside-plane ?O ?PL))
  (money -5) (time 15)))

(action (fly-plane ?PL ?L ?M)
 preconds ((a-is-a AIRPORT ?M)
  (c-at-plane ?PL ?L))
 effects ((c-at-plane ?PL ?M)
  (not (c-at-plane ?PL ?L))
  (money (* -15 (distance ?L ?M)))
  (time (* 5 (distance ?L ?M)))))

(action (move-truck-acities ?PL ?L ?M)
 preconds ((c-at-truck ?PL ?L))
 effects ((c-at-truck ?PL ?M)
  (not (c-at-truck ?PL ?L))
  (money (* -7 (distance ?L ?M)))
  (time (* 10 (distance ?L ?M)))))
```

Examples

Consider an example with the following initial state, goal and place positions. We show both the local and global rules that $PIPP_{GLOBAL}$ and $PIPP_{LOCAL}$ learn for only the first conflicting choice point.

Initial state =
(*A-IS-A AIRPORT AP*1) AP1*isanairport*
(A–IS–A AIRPORT AP2)
(A–SAME–CITY AP1 PO1)
(A–SAME–CITY PO1 AP1)
(A–SAME–CITY AP2 PO2)
(A–SAME–CITY PO2 AP2)
(C–AT–TRUCK TR1 AP1) *Truck*TR1*isat*AP1
(C–AT–TRUCK TRR AP2)
(C–AT–PLANE PL1 AP1)
(B–AT–OBJECT OB1 AP1) *Object*OB1*isat*AP1
(position AP1) = 15 (position AP2) = 21
(position PO1) = 13 (position PO2) = 22
(money 200) (time 0)
Goal=(B–AT–OBJECT OB1 PO2)

Suppose the system produces the following plan (we'll denote it by *P*1):

(*LOAD-PLANE OB*1 *PL*1 *AP*1)
(*FLY-PLANE PL*1 *AP*1 *AP*2)
(*UNLOAD-PLANE OB*1 *PL*1 *AP*2)
(*LOAD-TRUCK OB*1 *TRR AP*2)
(*MOVE-TRUCK-ACITIES TRR AP*2 *PO*2)
(*UNLOAD-TRUCK OB*1 *TRR PO*2)

and suppose the user inputs the following plan (let's call it *P*2):

(*LOAD-TRUCK OB*1 *TR*1 *AP*1)
(*MOVE-TRUCK-ACITIES TR*1 *AP*1 *AP*2)
(*UNLOAD-TRUCK OB*1 *TR*1 *AP*2)
(*LOAD-TRUCK OB*1 *TRR AP*2)
(*MOVE-TRUCK TRR AP*2 *PO*2)
(*UNLOAD-TRUCK OB*1 *TRR PO*2)

In the outcome of plan $P1$ the metric attributes have the following values (*money* 83) (*time* 56) and the metric attributes in the outcome of the plan $P2$ have the values of (*money* 123) (*time* 72). Then evaluating the quality function we get $q(P1) = 197$ and $q(P2) = 237$. Since $q(P2) > q(P1)$, $P2$ is labelled as a better plan and $P1$ as the worse plan. Having identified the better plan, we can calculate constraints on the range of values of the variables under which $P2$ is better than $P1$. This can be done by computing the values of metric attributes (i.e., money and time) in the final state by treating the variable parameters (i.e. the place positions) as symbols and subsituting thus computed symbolic values of money and time into the quality function and imposing the constraint $q(P2) > q(P1)$.

5-*wage* $\times$ (6 $\times$ (*distance PO*1 *AP*1) $-$ 20) $<$ -10 $\times$ (*distance PO*1 *AP*1),
where (*distance PO*1 *AP*1) = (*abs* (− (*position PO*1) (*position sAP*1))).

By solving the above inequality PIPP determines that *if* (*distance PO*1 *AP*1) $< \frac{100}{40} = 2.5$ *then* $P2$ *is better than* $P1$. This is the betterness condition of $P2$ over $P1$. Reversing the above inequality we get the betterness condition of $P1$ over $P2$. Under DerSNLP's scheme the external agent's plan will be stored with the relevant initial conditions

(*C-AT-TRUCK TR*1 *AP*1)
(*B-AT-OBJECT OB*1 *AP*2)
(*C-AT-TRUCK TR*1 *AP*1)
(*SAME-CITY AP*2 *PO*2)

and the goal condition (*B-AT-OBJECT OB*1 *PO*2) whereas the system's solution will be stored with the relevant initial conditions

(*C-AT-PLANE PL*1 *AP*1)
(*C-AT-TRUCK TR*1 *AP*1)
(*B-AT-OBJECT OB*1 *AP*2)

and the goal condition (*B-AT-OBJECT OB*1 *PO*2). Clearly, for a problem that has the union of the essential conditions of both these cases, the retrieving module will need to be able to decide as to which case to retrieve. In such cases, $PIPP_{GLOBAL}$ evaluates the betterness condition to discriminate between the cases.

$PIPP_{LOCAL}$ needs the *un-consumed* relevant conditions at the conflicting choice points to form the local rule. The first (and the only) point of conflict between the two plan-traces is the choice of (*UNLOAD-TRUCK OB*1 *TR*1 *AP*2) by the system's trace and (*UNLOAD-PLANE OB*1 *PL*1 *AP*2) by the external agent's trace to reduce the open-condition (*B-AT-OBJECT OB*1 *AP*2). The relevant initial conditions (*C-AT-TRUCK TRR AP*2) and (*SAME-CITY AP*2 *PO*2) are consumed by the action (*MOVE-TRUCK TRR AP*2 *PO*2) and hence do not need to be stored as antecedents for the local rule. This leaves

(*C-AT-TRUCK TR*1 *AP*1)
(*B-AT-OBJECT OB*1 *AP*2)
(*C-AT-PLANE PL*1 *AP*1))

as the un-consumed relevant conditions at this choice point. These conditions plus the betterness conditions of the planning choice of (*UNLOAD-PLANE OB*1 *AP*1), namely, (*distance AP*1 *AP*2) < 2.5, form the antecedent of the local rule (let's call it *LR*1).

if open-cond is (*B-AT-OBJECT OBx POy*))
and ((*C-AT-TRUCK TRx APx*)
(*B-AT-OBJECT OBx APy*)
(*C-AT-PLANEPlx APx*))
and (*distance Plx Ply*) < 2.5)) then
prefer (*UNLOAD-TRUCK OBx TRx PLy*)
over (*UNLOAD-PLANE OBx Px PLy*)))

Suppose that at this point $PIPP_{GLOBAL}$ is given the following problem to solve,

Initial state = (*B-AT-OBJ OB*10 *AP*10)
(*C-AT-TRUCK TR*10 *AP*10)
(*C-AT-PLANE PL*10 *AP*10)
(*A-IS-A AIRPORT AP*10)
(*A-IS-A AIRPORT AP*11)
(*SAME-CITY AP*10 *PO*10)
(*SAME-CITY PO*10 *AP*10)
(*SAME-CITY AP*11 *PO*11)
(*SAME-CITY PO*11 *AP*11)
(*C-AT-PLANE PL*11 *AP*11)
(*C-AT-TRUCK TR*12 *PO*11)
(*C-AT-TRUCK TR*11 *AP*11)
(*position AP*10) = 0 (*position AP*11) = 2
(*position PO*10) = 1 (*position PO*11) = 3
(*money*200) (*time*0)
Goal = (*B-AT-OBJ OB*10 *PO*11)

$PIPP_{GLOBAL}$ evaluates the antecedent of the only global rule it has learned. Since the new problem description has the union of the relevant initial conditions of both $P1$ and $P2$ and (*distance AP*10 *AP*11) < 2.5 PIPP, it retrieves the $P1$-'s trace (i.e., decision sequence that led to the truck-plan) and produce the following plan (let's call it $P3$).

(*LOAD-TRUCK OB*10 *TR*10 *AP*10)
(*MOVE-TRUCK-ACITIES TR*10 *AP*10 *AP*11)
(*UNLOAD-TRUCK OB*10 *TR*10 *AP*11)
(*LOAD-TRUCK OB*10 *TR*11 *AP*11)
(*MOVE-TRUCK TR*11 *AP*11 *PO*11)
(*UNLOAD-TRUCK OB*10 *TR*11 *PO*11)

Suppose the user inputs the following plan (let's call it $P4$).

(*LOAD-TRUCK OB*10 *TR*10 *AP*10)
(*MOVE-TRUCK-ACITIES TR*10 *AP*10 *PO*11)
(*UNLOAD-TRUCK OB*10 *TR*10 *PO*11)

PIPP determines the quality both plans and determines that $P4$ is better than $P3$. The betterness conditions for the betterness of $P4$ over $P3$ are

$30 \times (distance\ AP10\ PO11) + 10 \times$
$(distance\ AP10\ AP11) - 43 \times$
$(distance\ AP11\ PO11) < 140 \qquad (2)$

Next, suppose that the following problem is presented to PIPP.

$Initial\text{-}state = (B\text{-}AT\text{-}OBJ\ OB1\ AP1)$
$(C\text{-}AT\text{-}PLANE\ PL1\ AP1)$
$(C\text{-}AT\text{-}TRUCK\ TR1\ AP1)$
$(SAME\text{-}CITY\ AP2\ PO2)$
$(position\ AP1) = 0 \qquad (position\ AP2) = 10$
$(position\ PO1) = 2 \qquad (position\ PO2) = 11.5$
$Goal = (B\text{-}AT\text{-}OBJ\ OB1\ AP2)$

Since $LR1$ is applicable in this situation (because $(distance\ AP1\ AP2) > 2.5$), it guides PIPP to prefer the action $(UNLOAD\text{-}PLANE\ OB1\ PL1\ AP2)$ and to produce the plan:

$(LOAD\text{-}PLANE\ OB1\ PL1\ AP1)$
$(FLY\text{-}PLANE\ PL1\ AP1\ AP2)$
$(UNLOAD\text{-}PLANE\ OB1\ PL1\ AP2)$.

Evaluation

We have implemented the mechanism for learning global rules from the first conflicting choice point. We integrated this technique with DerSNLP+EBL and compared the performance of the new system (we call it $PIPP_{G1}$) with that of DerSNLP+EBL.

We randomly generated 100 problems from the modified logistics transportation domain described earlier. Each problem contained two trucks and one plane, which were distributed between two cities.

Training sets of 10, 20, 30 and 40 were randomly selected from the 100-problem corpus, and for each training set, the remaining problems served as the corresponding testing set. DerSNLP+EBL and $PIPP_{G1}$ were trained and tested on each of these four train-test problem sets. The metric of interest was the percentage of test problems in which $PIPP_{G1}$ produced a better-quality solution than did DerSNLP+EBL. For each of the test problems, the quality of the plan generated by PIPP was compared to the quality of the plan generated by DerSNLP+EBL. The quality was evaluated using the quality function described in the Domain section. We simply counted each case in which PIPP's plan had a higher quality value than DerSNLP+EBL's plan. The percentage of test problems on which this occurred is given in the first row of Table 1.

As the number of training problems increases, the percentage of higher-quality solutions generated by $PIPP_{G1}$ relative to DerSNLP+EBL increases. This is not too surprising, since DerSNLP+EBL is not designed to produced better-quality plans *per se*. At least, it was not designed to exploit metrics of the kind that $PIPP_{G1}$ has been designed to exploit. Note, however, that rows 2 and 3 in Table 1 provide running time data, and there is no significant cost disadvantage to $PIPP_{G1}$'s consideration of metric information.

$PIPP_{G1}$ does not produce worse quality solutions on any test example. However, the training and test problems had a low level of goal-interaction, i.e., there was only one package to move between cities. With the potential for more goal-interactions, we might expect $PIPP_{G1}$ to perform less well than DerSNLP+EBL on some problems. The reason is that the betterness conditions only guarantee that the better solution is better than the worse solution for solving the one goal independently. In solving that goal in conjunction with others, the worse plan may actually turn out to be better. The solution may be to use a strategy employed by DerSNLP+EBL and store all single goal planning episodes and only those multiple goal episodes for which the goals interact negatively. These are matters for further empirical investigation.

	number of training problems			
	10	20	30	40
number of $PIPP_{G1}$ plans better than DerSNLP+EBL plans	13/90 14%	18/80 23%	25/70 36%	22/60 37%
$PIPP_{G1}$ time (secs)	61.5	57	50.6	47.8
DerSNLP+EBL time (secs)	60.2	55.6	48.9	44.1

Table 1: Number of test problems for which $PIPP_{G1}$ generated a better quality plan than did DerSNLP+EBL, and running time for each system.

Related Research

Most of the early work on EBL (Mitchell, Keller, & Keddar-Cabelli 1986; Minton 1989; Etzioni 1990; Laird, Newell, & Rosenbloom 1987; Bhatnagar & Mostow 1994) can be considered as learning operational rules to make state-space problem solvers more efficient. Minton's (Minton 1989) PRODIGY/EBL learned control rules using explanation based learning. Bhatnagar and Mostow (Bhatnagar & Mostow 1994) designed FAILSAFE to learn control rules using EBL when the underlying domain theory was recursive. The objective of case-based reasoning system such as CHEF (Hammond 1990) and PRODIGY/ANALOGY (Veloso 1994) is also to speed up the problem solving process by remembering previous problem solving experiences and replaying them in a similar context. Most of these speed up learning systems are limited to learning from state-space planning systems and have difficulty analyzing the space of partial-order planners. However, recently both EBL (Kambhampati, Katukam, & Qu 1996) and derivational analogy techniques (Ihrig 1996) have been extended to learn from partial-order planners.

Kambhampati *et al.* (Kambhampati, Katukam, & Qu 1996) propose a technique based on EBL to learn control rules for partial-order planners and apply it to SNLP and UCPOP to learn rejection-rules. Rejection-

type rules are learned by generalizing the explanation of the planning failures. DerSNLP+EBL (Ihrig 1996) extends SNLP+EBL by learning from successes as well as failures. Using derivational analogy techniques, it remembers a past planning episode and replays its valid decisions when a similar problem arises again.

However, UCPOP+EBL, SNLP+EBL and DerSNLP+EBL do not aim at learning to improve the quality of their solutions and it is not clear how their explanation construction technique can be extended to provide explanation for the system's failure to produce better quality plans. Unlike UCPOP+EBL's and SNLP+EBL's failed partial plans, a lower quality plan does not contain any inconsistencies that can be regressed to compute the essential conditions that predict that taking the sequence of decisions of the failed case will lead to the failure.

The problem is that the autonomous speed up learning systems cannot learn operational quality rules because they are unable to recognize the learning opportunities (i.e., the better plans) when they arise. The plan quality measurement knowledge is required to (a) recognize a learning opportunity when it arises and (b) to analytically learn from this episode. Empirical learning, however, can be used by an apprenticeship system to learn operational rules by assuming that an external agent is always able to provide a better solutions. For instance, SCOPE (Estlin & Mooeny 1997) assumes that an omniscient expert is available to always provide the system with better solutions. For each of the planning-decisions points, it collects all the positive decisions (i.e., the user's decisions) and the negative decisions (i.e., the system's decisions). These positive and negative decisions are then passed to FOIL (Pazzani & Kibler 1992), an inductive/EBL concept learner, to induce the features of the partial plan whose presence is predictive of a positive decision. These concepts serve as operational quality rules that can be used during planning. Induction is certainly a good technique when no plan quality measurement knowledge is available but a sub-optimal strategy when such knowledge is available. In addition, because SCOPE does not incorporate any explicit representation of the quality goal into the search control rules, its rules can provide wrong guidance if the system's quality goals change.

Systems that possess quality knowledge

One reason that the problem of improving plan quality has been overlooked is the (often implicit) assumption that two separate phases of planning and scheduling have to be followed to solve most complex real world problems. The planner supplies a sequence of actions to achieve the desired goals and the scheduler assigns the resources and times to these actions optimizing some quality function of the resources. However, as attempts to automate the management of complex systems are made, it is becoming increasingly clear that the sharp division between planning and scheduling is neither always possible nor desirable (Wilkins & Desimone 1994; Tate, Drabble, & Kirby 1994; Muscettola 1994). Recently, some attempts have been made to integrate the two phases (Muscettola 1994) and/or to extend the classical planning framework to enable planners to reason about the resources and to optimize the resource consumption (Tate, Drabble, & Kirby 1994; Wilkins & Desimone 1994). Decision theoretic planners (such as PYRRHUS (Williamson 1996)) attempt to construct plans that optimize a decision value quality function. However, little work has been done to learn planning knowledge for such systems.

Perez (Perez 1996) describes QUALITY, a learning system, implemented on top of PRODIGY, that attempts to improve PRODIGY's control rules so that it can generate better quality plans. Given a problem and simple quality measurement knowledge, the system generates a plan and asks an expert to criticize it. If the expert's criticized solution is better than QUALITY's solution, according to the given metric, then QUALITY generates a trace of the user's plan and compares it with its own trace to identify the decision points where user's choice differs from the system's choice. It assigns a cost value to each node in user's and the system's planning traces using the given quality metric. QUALITY learns operational rules from those goal-nodes which have a zero cost in the user's trace and non-zero cost in the system's trace. A goal-node's cost can be zero because (a) it was true in the initial state (b) or because it was added as a side-effect of an operator chosen to achieve some other goal. The reason for the difference in the cost values of the two nodes is identified by examining both trees and the explanation thus constructed forms the antecedent of the control rule learned by QUALITY. Perez also suggests *control knowledge trees* to encode global knowledge.

Another learning method to improve plan quality, which also runs on PRODIGY, was proposed by Iwamoto (Iwamoto 1994). This technique also uses intra-solution learning to acquire control rules for near-optimal solutions in LSI design. It builds an explanation by backpropagating the weakest conditions, but excluding the conditions expressed in terms of predicates related to quality. Iwamoto's system only learns operator preference rules and does not learn binding and goal preference rules.

Both these systems are based on a state-space planner and only translate simple quality metrics. The approach taken in PIPP offers an improvement, because it can learn planning quality for partial-order planners, by operationalizing complex quality metrics.

Conclusion

To bridge the gap between theory and practice, AI planners need to be able to improve the quality of the plans they produce. Value-theoretic functions provide a well-developed framework for representing and reasoning with complex tradeoffs between various competing quality factors. This work reports a testbed system, PIPP, that we are developing to test the effectiveness

of using value-theoretic representation of plan quality to automatically acquire search control heuristics to generate better quality plans. The preliminary results are encouraging, particularly because this approach can be grounded in techniques used by real world planning researchers and operations research experts for representing and eliciting preference knowledge. Further empirical studies are in the works to evaluate its performance under a wider range of problem types.

References

Bhatnagar, N., and Mostow, J. 1994. On-line learning from search failures. *Machine Learning* 15:69–117.

Cohon, J. 1978. *Multiobjective programming and planning.* New York: Academic Press.

de Soete, G., and Hubert, F. 1990. *New developments in psychological choice modeling.* New York: North-Holland.

Drabble, B.; Gil, Y.; and Tate, A. 1995. Acquiring criteria for plan quality contro. In *proceedings of the 1995 AAAI Stanford Spring Symposium Workshop on Integrated Planning Applications.*

Ephrati, E.; Pollack, M.; and Milshtein, M. 1996. A cost-directed planner: Preliminary report. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1223–1228. Cambridge, MA: AAAI Press/MIT Press.

Estlin, T., and Mooeny, R. 1997. Learning to improve both efficiency and quality of planning. In *Proceedings of IJCAI.* Morgan Kaufmann.

Etzioni, O. 1990. A structural theory of explanation based learning. Technical Report CMU-CS-90-185, PhD Thesis, Carnegie Mellon University.

Fishburn, P. 1970. *Utility theory for decision making.* New York: Wiley.

Haddawy, P., and Hanks, S. 1998. Utility models for goal-directed, decision-theoretic planners. *omputational Intelligence* 14(3).

Hammond, K. 1990. Case-based planning: A framework for planning from experience. *Cognitive Science* 14(3).

Hensher, D. 1981. *Applied discrete-choice modelling.* New York: Wiley.

Ihrig, L., and Kambhampati, S. 1997. Storing and indexing plan derivations through explanation-based analysis of retrieval failures. *Journal of Artificial Intelligence Research* 7:161–198.

Ihrig, L. 1996. The design and implementation of a case-based planning framework within a partial order planner. Technical Report ASU-CSE-96-007, PhD thesis, Department of Computer Science, Arizona State University.

Iwamoto, M. 1994. A planner with quality goal and its speed up learning for optimization problems. In *Proceedings of Second International Conference on AI Planning Systems, Chicago, IL*, 281–286.

Kambhampati, S.; Katukam, S.; and Qu, Y. 1996. Failure driven dynamic search control for partial order planners. *Artificial Intelligence* 88:253–316.

Keeney, R., and Raiffa, H. 1993. *Decisions with multiple objectives : preferences and value tradeoffs.* New York: Cambridge University Press, 2nd edition.

Laird, J.; Newell, A.; and Rosenbloom, P. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(3).

Louviere, J. 1988. *Analyzing decision making : metric conjoint analysis.* Newbury Park: Sage Publications.

Minton, S. 1989. Expalantion-based learning. *Artificial Intelligence* 40:63–118.

Mitchell, T.; Keller, R.; and Keddar-Cabelli, S. 1986. Explanation based learning: A unifying view. *Machine Learning* 1:47–80.

Muscettola, N. 1994. HSTS: Integrating planning and scheduling. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling.* San Francisco: Morgan Kaufmann. 169–212.

Pazzani, M., and Kibler, D. 1992. The utility of background knowledge in inductive learning. *Machine Learning* 9:57–94.

Perez, A. 1996. Representing and learning quality-improving search control knowledge. In Saitta, L., ed., *Proceedings of the Thirteenth International Conference on Machine Learning.* Los Altos, CA: Morgan Kaufmann.

Ruby, D., and Kibler, D. 1993. Learning steppingstones for problem solving. *International Journal of Pattern Recognition and Artificial Intelligence* 7(3):527–540.

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-Plan2: An open architecture for command, planning and control. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling.* San Francisco: Morgan Kaufmann. 213–240.

Veloso, M. 1994. *Learning by Analogical Reasoning.* Berlin: Springer Verlag.

Wilkins, D., and Desimone, R. 1994. Applying an AI planner to military operations planning. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling.* San Francisco: Morgan Kaufmann. 685–710.

Wilkins, D. C. 1988. Knowledge base refinment using apprenticeship learning technques. In *Proceedings of the National Conference on Artificial Intelligence*, 646–651. Menlo Park, CA: AAAI Press.

Williamson, M. 1996. A value-directed approach to planning. Technical Report TR-96-06-03, PhD thesis, University of Washington.

Yu, P. 1985. *Multiple-criteria decision making : concepts, techniques, and extensions.* New York: Plenum Press.