Lecture 23: Learning - 6 & Resource Bounded Reasoning -1

Victor Lesser

CMPSCI 683
Fall 2004

Today's lecture

- Instance-Based Learning
- Analytical Learning (Explanation-Based Learning)
  - First work done at Umass on learning rules of Baseball
  - Material take from Mitchell's Machine Learning
- Begin Resource Bounded Reasoning

Instance-Based Learning

- Encode specific experiences in memory rather than abstractions
- Carry out generalizations at the retrieval time rather than the storage time -- *lazy learning*.
- In their most general form:
  - Based on partial match on a similarity metric, retrieve a set of cases/instances most "relevant" to the present context.
  - Adapt the retrieved experiences to new situations. This could be based on algorithms ranging from a simple k-nearest neighbor classification to chains of reasoning.

Example of Instance-Based Learning

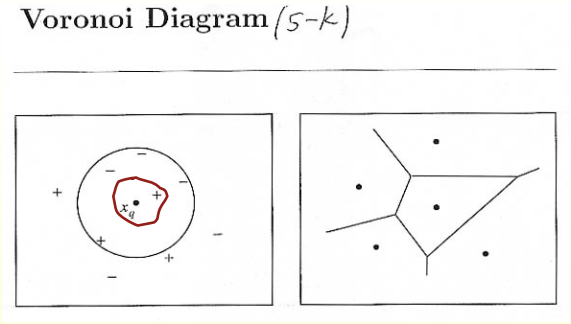
- Key idea: just store all training examples $\langle x_i, f(x_i)\rangle$
- Nearest neighbor:
  - Given query instance x_q, first locate nearest training example x_n, then estimate

$$\hat{f}(x_q) \leftarrow f(x_n)$$

- *K*- Nearest neighbor
  - Given x_q, take vote among its *k* nearest neighbors (if discrete-valued target function)
  - Take mean of *f* values of *k* nearest neighbors (if real-valued)

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Voronoi Diagrams



On left, Positive and Negative Training Examples

- Nearest neighbor yes for x_q, 5-k classifies it as negative

On right, is decision surface for nearest neighbor, query in region will have same value

Distance-Weighted *k*NN

- **Might want weight nearer neighbors more heavily…**

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

- **Where**

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

- **And $d(x_q, x_i)$ is distance between x_q, x_i**
- **Note now it makes sense to use all training examples instead of just *k***
  - *Classification much slower*

Curse of Dimensionality

- Imagine instances described by 20 attributes, but only 2 are relevant to target function
- *Curse of dimensionality*: nearest neighbor is easily mislead when high-dimensional X
  - Similar to overfitting
- One approach:
  - Stretch j^{th} axis by weight z_j, where $z_1, \ldots, z_n$ chosen to minimize prediction error
    - Length the axes that correspond to the more relevant attributes
  - Use cross-validation to automatically choose weights $z_1, \ldots, z_n$
    - Minimize error in classification
    - Setting z_j to zero eliminates this dimension all together

When to Consider Nearest Neighbor

- **Instances map to points in $\Re^n$**
  - Continuous real values
- **Less than 20 attributes per instance**
- **Lots of training data**
- **Advantages:**
  - Training is very fast
  - Robust to noise training data
  - Learn complex target functions
  - Don't lose information
- **Disadvantages:**
  - Slow at query time
  - Easily fooled by irrelevant attributes

Locally Weighted Regression: Approximating Real-Valued Function

- **Note *k*NN forms local approximation to *f* for each query point x_q**
- **Why not form an explicit approximation $\hat{f}(x)$ for region surrounding x_q**
  - Fit linear function to *k* nearest neighbors
    - $F(x)= w_0+w_1a_1+ \ldots\ldots w_na_n$
  - Fit quadratic….
  - Produces "piecewise approximation" to *f*
- **Several choices of error to minimize -- exploit grading descent**
  - Squared error over *k* nearest to neighbors

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \text{ in } k \text{ nearest nbrs of } xq} (f(x) - \hat{f}(x))^2$$

  - Distance-weighted squared error over all neighbors
  - …

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

Case-Based Reasoning

- **Can apply instance-based learning even when $X \neq \Re^n$**
- **Need different distance metric**
- **Case-Based Reasoning is instance-based learning applied to instances with symbolic logic descriptions**

```
((user-complaint error53-on-shutdown)
 (cpu-model PowerPC)
 (operating-system Windows)
 (network-connection PCIA)
 (memory 48meg)
 (installed-applications Excel Netscape VirusScan
 (disk 1gig)
 (likely-cause ???) )
```

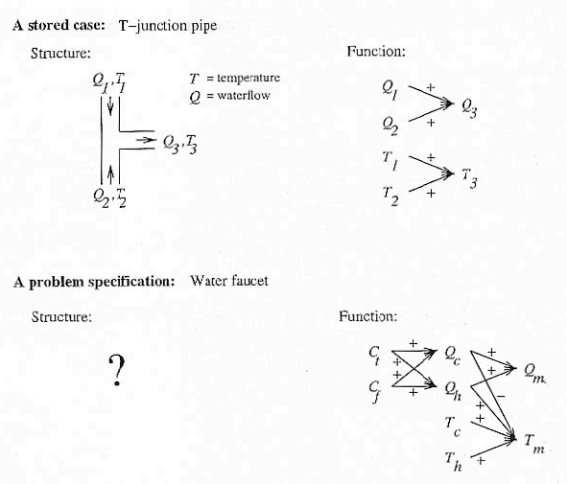

Ingredients of Problem-solving CBR

- **Key elements of problem solving CBR are:**
  - Cases represented as solved problems
  - Index cases under goals satisfied and planning problems avoided
  - Retrieve prior case sharing the most goals & avoiding the most problems
  - Adapt solution of prior case to solve a new case. May require re-solving problems and/or repairing solutions
  - Index new case and solution under goals satisfied and planning problems avoided

Case-Based Reasoning in CADET

- **CADET: 75 stored examples of mechanical devices**
  - Each training example: [qualitative function, mechanical structure]
  - New query: desired function
  - Target value: mechanical structure for this function
- **Distance metric: match qualitative function descriptions**
  - Size of largest subgraph between two function graphs

Case-Based Reasoning in CADET



Exploits domian specific rewrite rule to modify cases to make matching more likely

A + --> B

to

A + --> x + --> B

Case-Based Reasoning in Chef

- **CHEF consists of six processes:**
  - **Problem anticipation**: the planner anticipates planning problems by noticing features in the current input that have previously participated in past planning problems
  - **Plan retrieval**: The planner searches for a plan that satisfies as many of its current goals as possible while avoiding the problems that it has predicted
  - **Plan modification**: The planner alerts the plans it has found to satisfy any goals from the input that are not already achieved
  - **Plan repair**: When a plan fails, the planner fixes the faculty plan by building up a casual explanation of why the failure has occurred and using it to find the different strategies for repairing it
  - **Credit assignment**: Along with repairing a failed plan, the planner wants to repair the characterization of the world that allowed it to create the failed plan in the first place. It does this by using the casual explanation of why the failure occurred to identify the features in the input that led to the problem and then mark them as predictive of it
  - **Plan storage**: The planner places successful plans in memory, indexed by the goals that they satisfy and the problems that they avoid

Beef-with-green-beans

A half pound of beef
Two tablespoons of soy sauce
One teaspoon of rice wine
A half tablespoon of corn starch
One teaspoon of sugar
A half pound of green bean
One teaspoon of salt
One chunk of garlic

- **Chop the garlic into pieces the size of matchheads**
- **Shred the beef**
- **Marinate the beef in the garlic, sugar, corn starch, rice wine and soy sauce**
- **Stir fry the spices, rice wine and beef for one minute**
- **Add the green bean to the spices, rice wine and beef**
- **Stir fry the spices, rice wine, green bean and beef for three minutes**
- **Add the salt to the spices, rice wine, green bean and beef**

Request for a New Recipe

Recipe for BEEF-AND-BROCOLI
Found nearest recipe is BEEF-WITH-GREEN-BEANS

Modifying recipe: BEEF-WITH-GREEN-BEANS
To satisfy: include broccoli in the dish

Placing some broccoli in recipe BEEF-WITH-GREEN-BEANS

-Considering ingredient-critic:
Before doing step: Stir fry the -Variable-
do: Chop the broccoli into pieces the size of chunks
-Ingredient critic applied

Chef alters old plans to satisfy new goals using a set of modification rules and a set of new objects

Check Whether New Recipe Works via Simulation

A half pound of beef
Two tablespoons of soy sauce
broccoli
One teaspoon of rice wine
A half tablespoon of corn starch
One teaspoon of sugar
A half pound of
One teaspoon of salt
One chunk of garlic

- **Chop the garlic into pieces the size of matchheads**
- **Shred the beef**
- **Marinate the beef in the garlic, sugar, corn starch, rice wine and soy sauce**
- **Chop the broccoli into pieces the size of chunks**
- **Stir fry the spices, rice wine and beef for one minute**
- **Add the broccoli to the spices, rice wine and beef**
- **Stir fry the spices, rice wine, broccoli and beef for three minutes**
- **Add the salt to the spices, rice wine, broccoli and beef**

The beef is now tender.
The dish now tastes savory.
The broccoli is not crisp.
The dish now tastes salty.
The dish now tastes sweet.
The dish now tastes like garlic.

Plan Repair

- ALTER-PLAN:SIDE-EFFECT: Replace the step that causes the violating condition with one that does not have the same side-effect but achieves the same goal
- SLPIT-AND-REFORM: Split the step into two separate steps and run them independently
- ADJUNT-PLAN:REMOVE: Add a new step to be run along with a step that causes a side-effect that removes the side-effect as it is created

Plan Storage

Indexing BEEF-AND-BROCCOLI under goals and problems:

If this plan is successful, the following should be true:
- The beef is now tender.
- The broccoli is now crisp.
- Include beef in the dish.
- Include broccoli in the dish.
- Make a stir-fry dish.

The plan avoids failure exemplified by the state
“The broccoli is now soggy” in recipe BEEF-AND-BROCCOLI.

Problem Anticipation

Searching for plan that satisfies input goals-
- Include chicken in the dish.
- Include snow pea in the dish.
- Make a stir-fry dish.

Collecting and activating tests.
- Is the dish STYLE-STIR-FRY
- Is the item a MEAT
- Is the item a VEGETABLE
- Is the TEXTURE of item CRISP

Chicken+Snow pea+Stir Frying= Failure
“Meat sweats when it is stir-fried.”
“Stir-frying in too much liquid makes crisp vegetables soggy.”
Reminded of a failure in the BEEF-AND-BROCCOLI plan.
Failure= “The vegetable is now soggy”

Plan Retrieval

Driving down on: Make a stir-fry dish.
Succeeded-
Driving down on:
Avoid failure exemplified by the state "The broccoli is now soggy" in recipe BEEF-AND-BROCCOLI
Succeeded
Driving down on: Include chicken in the dish
Failed- Trying more general goal
Driving down on: Include meat in the dish
Succeeded
Driving down on: Include snow pea in the dish
Failed-Trying more general goal
Driving down on: Include vegetable in the dish.
Succeeded

Found recipe→ REC9 BEEF-AND-BROCCOLI

The Inductive Generalization Process

- **Given**
  - Space of Hypotheses
  - Training examples of target concept
- **Determine**
  - Hypothesis consistent with the training examples

May need a lot of training examples to distinguish relevant from irrelevant features

The Analytical Generalization Problem

- **Give**
  - Space of Hypotheses
  - Training examples of target concepts
  - *Domain theory for explaining examples*
- **Determine**
  - Hypothesis consistent with *both* the training examples *and the domain theory*

May need fewer training examples to distinguish relevant attributes because generalization can be based on logical rather than statistical reasoning

Why Analytical Learning

- **Normal Learning**
  - Involves a long process of uncovering the consequences of prior knowledge
  - Guided by a specific training examples
- **Use prior knowledge to reduce the complexity of the hypothesis space to be searched**
  - Reducing sample complexity
  - Improving generalization accuracy

An Analytical Generalization Problem

Given:

- **Instances: pairs of objects**
  - represented by the predicates Type, Color, Volume, Owner, Material, Density and On.
- **Hypotheses: sets first-order if-then rules - horn clauses**
- **Target Concept:** ***Safe-to-stack(x,y) -- this is what needs to be learned***
- **Training Example: Safe-to-stack(OBJ1,OBJ2)**

On(OBJ1,OBJ2)
Type(OBJ1,BOX)
Type(OBJ2,ENDTABLE)
Color(OBJ1,RED)
Color(OBJ2,BLUE)
Volume(OBJ1,.1)
Density(OBJ1,.1)
…

An Analytical Generalization Problem, *cont'd*

- **Domain Theory:**

Safe-To-Stack(x,y) <- Not(Fragile(y))
Safe-To-Stack(x,y) <- Lighter(x,y)
Lighter(x,y) <- Weight(x,wx) and Weight(y,wy) and Less(wx,wy)
Weight(x,w) <- Volume(x,v) and Density(x,d) and Equal(w,v*d)
Weight(x,5) <- Type(x, ENDTABLE)
…

Determine: Hypothesis (h) consistent with domain theory (B) if B does not entail not (h)

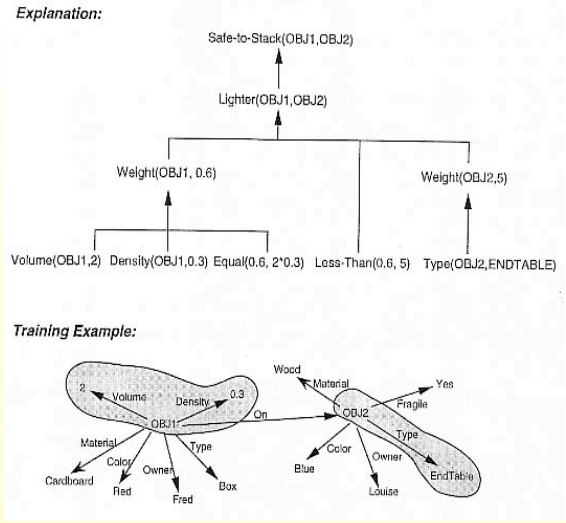
Prolog EBG

Initialize hypothesis = {}

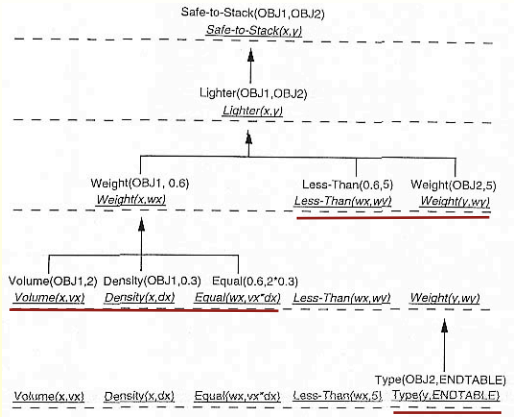
- **For each positive training example not covered by hypothesis:**

1. ***Explain*** **how training example satisfies target concept, in terms of domain theory**
2. ***Analyze*** **the explanation to determine the most general conditions under which this explanation (proof) holds**
3. ***Refine*** **the hypothesis by adding a new rule, whose preconditions are the above conditions, and whose consequent asserts the target concept**

Explanation of a Training Example



Computing the Weakest Preimage of Explanation



Regression Algorithm

Regress(*Frontier, Rule, Expression, $U_{I,R}$*)

Frontier: the set of expressions to be regressed through *Rule*
Rule: a horn clause.
Expression: the member of *Frontier* that is inferred by *Rule* in the explanation.
$U_{I,R}$: the substitution that unifies *Rule* to the training example in the explanation

Returns the list of expressions forming the weakest preimage of *Frontier* with respect to *Rule*

let Consequent ← *Rule* consequent
let Antecedents ← *Rule* antecedents
1. $U_{E,R}$ ← most general unifier of *Expression* with Consequent such that there exists a substitution S for which

$S(U_{E,R}(\text{Consequent})) = U_{I,R}(\text{Consequent})$

2. Return $U_{E,R}(\{Frontier\text{-}Consequent+Antecedent\})$

Example:

Regress({Volume(x,vs), Density(x,dx), Equal(wx,vx*dx), Less-Than(wx,wy), Weight(y,wy)), Weight(z,5) :- Type(z,ENDTABLE), Weight(y,wy), {OBJ2/z})

Consequent ← Weight(z,5)
Antecedents ← Type(z,ENDTABLE)
$U_{E,R}$ ← {y/z, 5/wy}, (S = {OBJ2/y})

Result ← {Volume(x,vs), Density(x,dx), Equal(wx,vx*dx), Less-Than(wx,5), Type(y,ENDTABLE)}

New Rule

SafeToStack(x,y) <--

Volume(x,vx) & Density(x,dx) & Equal(wx,times(vx,dx) & LessThan(wx,5) & Type(y,Endtable)

Lessons from Safe-to-Stack Example

- **Justified generalization from single example**
  - Compiling relevant parts of domain theory
- **Explanation determines feature relevance**
- **Regression determines needed feature constraints**
- **Generality of result depends on domain theory**
- **Still require multiple examples**

Perspectives on Anayltical Learning

- **Theory-guided generalization from examples**
- **Example-guided operationalization of theories**
- **"Just" restating what learner already "knows"**

Is it learning?

- **Are you learning when you get better over time at chess?**
  - Even though you already know everything in principle, once you know rules of the game…
- **Are you learning when you sit in a math class?**
  - Even though those theorems follow deductively from the axioms you've already learned...

Summary of Explanation-Based Learning

- Knowledge of the domain is represented as logical rules
- View learning as a form of reasoning
- Background knowledge is used to construct proofs or 'explanations of experience'
- These proofs are then "generalized" (based on a type of goal regression) and compiled into rules
- It is a process of transforming existing knowledge into another form so as to be able to apply it efficiently – also called speedup learning

Summary of Learning

- Learning as a search for the "best" hypothesis/function that matches the training data
- Learning technology has become very formal and closely related to statistics
- Credit Assignment is a complex problem
- Many different approaches to learning based on
  - Supervised vs. Unsupervised
  - Learning Single step decision process vs. multiple step one
  - On-line/Incremental vs. Off-line learning
  - Character of Function that needs to be learned
  - Quality and Character of Training Data
- We have just scratched the surface
  - Support Vector Machines, Hidden Markov Models, Baysean Network Learning, …
  - Field is evolving very quickly

Resource Bounded Reasoning

Problem: Agents cannot be perfectly rational

- Agents have limited computational power.
- They must react within an acceptable time.
- Computation time delays action and reduces the value of the result.
- Must cope with uncertainty and missing information.
- Limited planning horizon.
- The "appropriate" level of deliberation is situation dependent.

It is beneficial to build systems that can tradeoff computational resource for quality of results.

More on the Problem

- **AI deliberation techniques have a large variability in execution time, memory consumption, etc.**
  - Difficult to predict reasonable upper bounds:
    - Non-deterministic search
    - Variable amount of data
    - Variable amount of uncertainty
- **The computational resources required to reach an optimal decision normally reduce the overall utility of the result.**

Approach

- **Not only an issue of more computing power.**
  - Need intelligent control
- **How to build agents that "satisfice" rather than "optimize"?**
  - How to get the "best" answer within available resources

Satisficing

- **Proposed by Herbert Simon in 1957 to denote decision making that searches until an alternative is found that meets the agent's aspiration level criterion.**

"It appears probable that, however adaptive the behavior of organisms in learning and choice situations, this adaptiveness falls far short of the ideal "maximizing" postulated in economic theory. Evidently, organisms adapt well enough to "satisfice"; they do not, in general, "optimize"."

Problem Characteristics

- **It is not feasible (computationally) or desirable (economically) to compute the optimal answer**
- **The "appropriate" level of deliberation is *situation dependent***
- **It is beneficial to build systems that can tradeoff computational resources for quality of results**

Key to Applying AI in Open Environments

Computational/Resources Tradeoffs

- **Computation Time versus Quality of Results**
- **Memory versus Time versus Power**
- **Cost of Information versus Quality of Results**
- **Communication Bandwidth versus Quality**
- **Many more …….**

Approaches to Satisficing

- Satisficing = **approximate reasoning**
- Satisficing = **approximate modeling**
- Satisficing = **approximate meta-reasoning**
- Satisficing = **bounded optimality**
- Satisficing = **a combination of the above**

An Example of Satisficing : Vehicle Monitoring

- **Goal**
  - Within 10 seconds, supply the vehicle types, positions, and movement characteristics for as many vehicles moving through the environment as possible, giving preference to tracking vehicles of type v1 and determining vehicle directions with high precision.

(v1, s1, l1, d1) (v2, s2, l2, d2)

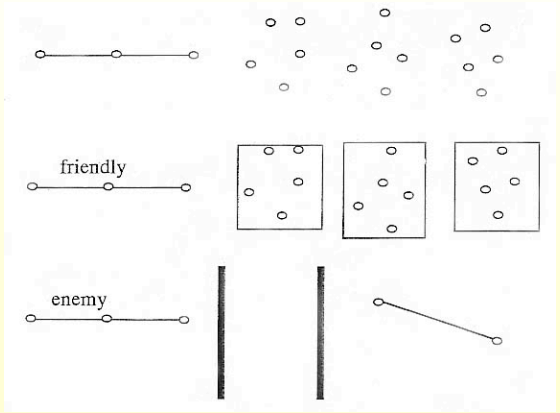
- **Best possible**
  - The following events and their corresponding features are certain: Vehicle type v1 located at l1, moving with speed s1 and direction d1. Vehicle type v2 at 12 moving with speed s2 and direction d2.

    No other vehicles are present in the environment.
- **Best within Deadline:**
  - From a cursory analysis, it is likely that there exists a vehicle of type v1 located near l1, moving with speed between s0 and s1 in direction d1.

    Other vehicles might be present.

Situation Dependent Satisficing

- Appropriate approximation is based both on the needs of the user and on the current state



Applications

- Medical diagnosis of treatments
- Combinatorial Optimization
- Evaluation of belief networks
- Real-time heuristic search
- Information Gathering
- Mobile robot navigation
- Database query optimization
- Software validation
- Real-time Graphics

Real-Time Graphics

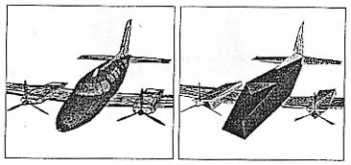


Fig. 1: ***Flexible strategy for reducing the complexity of models.*** Methods for reducing the number of vertices used to describe a geometric model can be used to generate a spectrum of models from an ideal model to progressively simpler models. The fully detailed model (left) is described by 6,795 vertices. The simplified model (right) is described by 97 vertices.

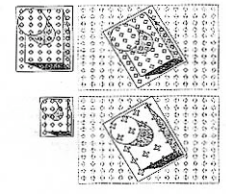


Fig. 3: ***Reducing the spatial solution.*** Objects can be sampled at a variety of spatial resolutions before being composited into final scenes, allowing for a trade-off between the clarity of an object and the computation required to render the object.

Horvitz & Lengyel: *Working Notes of the AAAI Fall Symp. on Flexible Computation, 1996*

Real Time Graphics, cont.

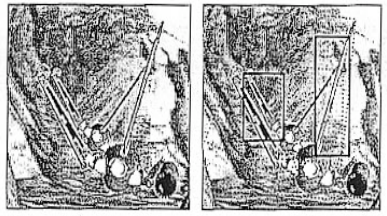


Fig. 4: ***Minimizing computation via sprite reuse.*** Talisman takes advantage of spatial and temporal coherence by transforming previously rendered sprites instead of re-rendering the sprites. Rectangles composed of dotted lines bound sprites warped to fit a new frame; rectangles of solid lines bound re-rendered sprites.

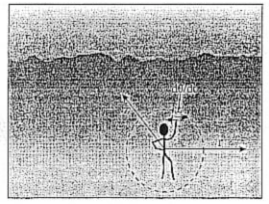


Fig. 8: ***Role of visual attention.*** We seek to understand the variation of the sensitivity of viewers to various classes of graphics degradation as spatial and temporal relationships change. This figure highlights opportunities for modeling attention as functions of the primary focus of attention and such relationships as adjacency (r), distance in the background (z), and motion of sprites $\left(\frac{d\theta}{dt}\right)$.

Horvitz & Lengyel: *Working Notes of the AAAI Fall Symp. on Flexible Computation, 1996*

Resource-bounded reasoning

A methodology for building satisficing systems by addressing the following four major issues:

1. **Elementary algorithm construction**
2. **Performance measurement and prediction**
3. **Composability of methods (subsystems)**
4. **Monitoring and meta-level control**

In the context of an Overall System Architecture

Elementary Algorithm Construction

- Develop computational methods that allow small quantities of resources - such as time, memory, or information - to be traded for gains in the value of computed results.
- Quality measures replace "correctness"
  - **Certainty** - Likelihood that the answer is correct.
  - **Precision** - Accuracy of the answer.
  - **Specificity** - Level of detail in the answer.
  - **Completeness** - Part of problem solved
  - **Cost** - Overall solution cost.
  - **Multidimensional quality measures.**

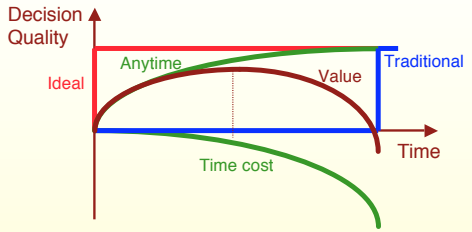
Principles of Meta-Level Control

- What's are the base-level computational methods?
- How does the system represent and project resource/quality tradeoffs?
- What kind of meta-level control is used?
- What are the characteristics of the overall system?

Anytime Algorithms

- An anytime algorithm is an algorithm whose quality of results improves gradually as computation time increases
- Anytime algorithms have been designed for planning, Bayesian inference, CSPs, combinatorial optimization, diagnosis

Anytime Algorithms



- Ideal (maximal quality in no time)
- Traditional (quality maximizing)
- Anytime (utility maximizing)

Two Types of Anytime Algorithms

- **Interruptible algorithms are anytime algorithms whose run time need not be determined in advance**
  - They can be interrupted at any time during execution and return a result
- **Contract algorithms are anytime algorithms that take the deadline as input**
  - No assumption is made about the results produced before the deadline

Performance Profiles

- **Both interruptible and contract algorithms have performance profiles, $Q(t)$, which return quality as a function of time**
- **Note that for contract algorithms the horizontal axis is the time that was given *in advance***

Contract Algorithms

- **Good contract algorithms can be easier to design because they can optimize with respect to the extra time input**

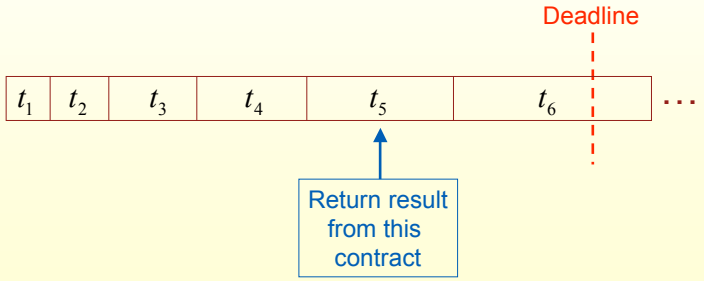
  Examples:
- **Depth-bounded or cost-bounded tree search**
- **Discretizing a continuous problem**
- **Composition of anytime algorithms**

Contract → Interruptible

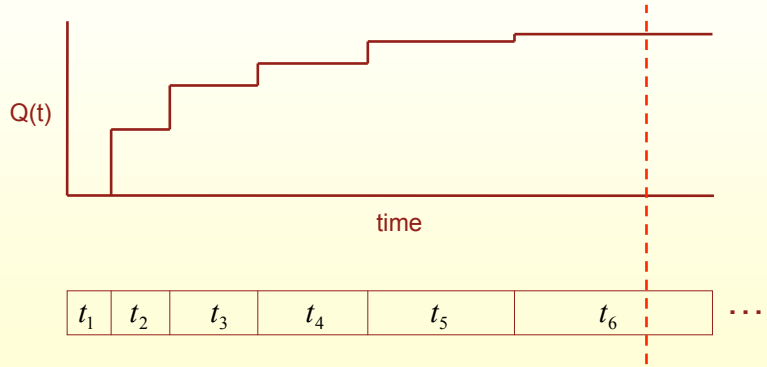
- **What if we want to use a contract algorithm in a setting where we don't know the deadline?**
- **We can repeatedly activate the contract algorithm with increasing run times**

Contract → Interruptible

- When the deadline occurs, we can return the result produced by the last contract to finish:



The Resulting Performance Profile



Next Lecture

- More on Resource Bounded Reasoning