Some Basics for Search Methods

Search paradigms

Some representation paradigms

SEARCH PARADIGMS

Breadth-first

Expanding the search tree chronologously, level by level

Depth-first

Expanding the search tree chronologously, branch by branch

Best-first

Expanding the search tree opportunistically, at position considered "best"

BREADTH-FIRST SEARCH

Properties

No heuristic knowledge required

Always finds optimal solution

Exponential complexity

Exponential storage requirement

Use

Frequent use, but in highly modified forms

Speech analysis, syntactic processing, machine translation, …

DEPTH-FIRST SEARCH

Properties

Heuristic knowledge exploitable

Finding a solution depends on choice of depth (none, optimal, suboptimal)

Complexity depends on exploitation of infomation

(cut-offs possible, depending on properties of evaluation scores)

Linear storage requirement

Ordering of expansions may be essential, if cut-offs possible

Iterative deepening possible

Use

Frequent use in modified forms, some use within homogeneous representations

Compound processes, constraint systems

CHRONOLOGOUS SEARCHES

Extensions

Combination of breadth- and depth-first

Speculative search space limitations

- **Limiting the number of alternatives considered (*beam search*)**
- **No guarantee of optimal solution**
- **No solution guarantee at all**
- **Large search spaces with reasonable intermediate information manageable**

Use

Beam search very widely used

Combinations for compound processes, according to domain preferences

BEST-FIRST SEARCH (A*)

Properties

Homogenous evaluation of problem states required

Concept of optimal path costs: $f^*(n) = g^*(n) + h^*(n)$

Heuristic estimates of optimal path costs: $f(n) = g(n) + h(n)$

- **$g(n)$ … minimal path costs found to current state**
- **$h(n)$ … estimated path costs from current state to goal state**

Node associated with best heuristic score is expanded next

Theorem: If $h(n) \leq h^*(n)\ \forall\ n$, A* is *admissible* (finds optimal solution)

Use

Machine translation

Specific subprocesses

EXAMPLE - BEST-FIRST SEARCH (GENERATING REFERRING EXPRESSIONS)

Problem description

Identifying *a set* of referents with *one* expression

Accumulating descriptors, including *boolean combinations*

Search space modeling

Initial state – empty description

Goal state – a distinguishing description (some are optimal)

Intermediate state – a non-distinguishing description

Search space exploration – Description expansion

***All* intermediate results can be expanded further – these are the nodes**

(only the full expression in the incremental algorithm)

EXAMPLE - BEST-FIRST SEARCH (II)

Evaluation function

n … node associated with some partial description

f(n) = (minimal) number of descriptors for a distinguishing description

g(n) = number of descriptors used in partial description generated so far in n

h(n) = complexity of descriptor combinations still unused at state n

Search tree expansion

Build successors of node evaluated best, if none yet unexplored,

with minimally complex descriptor combination still available there

Choose as the next node the one among open ones with best score that have

- minimum number of potential distractors still to be excluded

- minimum complexity of descriptor combinations still unused

SEARCH OPTIMIZATIONS - CUT-OFFS (I) (EXPLOITING ADMISSIBILITY)

Effect of admissibility

Algorithm terminates if no more nodes are worth expanding

(heuristic score is not better than proved value of a solution)

Value Cut-off – Realization of consequence of admissibility

If a solution is found (f(n)), its score is compared to each node

If for that node partial score plus the optimistic estimate is below the score of the best solution found so far ($g(n_i) + h(n_i) \geq f(n)$)

Then this node is closed

SEARCH OPTIMIZATIONS - CUT-OFFS (II)
JUSTIFIED BY GOAL REQUIREMENTS OR ASSUMPTIONS

Dominance Cut-off

Applicable to – Sibling nodes with

- **partial descriptions excluding the same potential distractors**
- **the same set of descriptors available**

The node with partial description evaluated worse is closed (worse g(n))

Complexity Cut-off

Description considered too complex not generated, may not yield a solution

Expression Cut-off

“Mixed” disjunctions excluded (“car or red”), also may not yield a solution

ASSUMPTIONS

Value Cut-off

Descriptors map one-to-one onto surface expression components

Modification of simple counting possible (but must still be monotone)

Dominance Cut-off

Compositionality of expressions

Complexity Cut-off

Complex expressions impractical, task modification required

Partitioning the identification task, focus narrowing, then identification

Expression Cut-off

“Mixed” disjunctions impractical, task modification required

Partitioning the set of intended referents for separate identification

COMPARING CUT-OFF MEASURES

	cut-off measures			
	all	*value*	*dominance*	*complexity*
avg. ***search tree size***	**2.2**	**3.88**	**2.33**	**61.64**
max. ***search tree size***	**9**	**71**	**11**	**945**
avg. ***search time (msec)***	**127.7**	**168.1**	**595.0**	**1133.1**
max. ***search time (msec)***	**690**	**2320**	**4550**	**19210**

REPRESENTATION AS RULES

Components of a rule

The precondition (if *…)* determines the applicability of a rule

The postcondition/action (then *…)* determines derived knowledge (applicable actions)

There are two types of rules (distinction essential for commutative rule systems):

- ***Implications* (*deduction*), deriving the truth of a fact**
- ***Actions*, which change a state**

Examples

Grammar rules

Lexical correspondences (in machine translation)

REPRESENTATION AS CONSTRAINTS

Components of a constraint

- **A set of variables**
- **A set of values relating variables to each other**

Constraint problem

Solutions to a constraint problem are assignments of values to variables so that all constraints are fulfilled

Examples

Equations (about structure sharing) in unification grammars

Collocation constraints

COMPARISON BETWEEN RULES AND CONSTRAINTS

Evaluation of information

- **Rules are directed, constraints not (that is, rules are more general)**
- **Restricted evaluation potential for rules**

Efficiency

Restricted interpretation enables more efficient evaluation for rules

Advantages/disadvantages

- **Rules are more modular and easier to adapt**
- **Constraints enable better information evaluation**
- **Both are associated with a flow of control that is difficult to understand**

Application areas

- **Rules suited for domain with isolated knowledge – control over application**
- **Constraints suited for coherent theories – constraint solver as a black box system**

EXAMPLE - CONSTRAINT SYSTEMS (Gardent 2002) (GENERATING REFERRING EXPRESSIONS)

Problem description

Identifying *a set* of referents (*S*) with *one* expression *L*

Accumulating descriptors, including *boolean combinations*

Problem modeling (truth (1+2) and contribution (3))

1. All properties applicable to *S* $\supseteq$ all properties in *L*

2. All properties not applicable to *S* $\supseteq$ all negative properties in *L*

3. For all distractors *C* of *S*: properties of *S* other than those of *C* > 0 or
non-properties of *S* but properties of *C* > 0

Extensions for disjunctions

A disjunction is a distinguishing description for a set of individuals *S*
if there is an element in the disjunction identifying covering subsets of *S*

EXAMPLE - CONSTRAINT SYSTEMS (II)

Distribution strategy (how to assign values to variables)

Case distinctions over cardinality of *L*, starting with minimal value

Algorithm stops once a solution is found

Implementation and results

Concurrent programming language Oz (PSE, Uni SB, 1998)

Supports set variables ranging over finite sets of integers

Runtime example: “the poodle, the jersey, the pitbul, and the poodle”

(10 objects, 10 descriptors, sparsely attributed) 1,4 sec

ATTACKING DEFICITS OF THE INCREMENTAL ALGORITHM – REDUNDANCY (see Gardent 2002)

descriptors/objects	x_1	x_2	x_3	x_4	x_5	x_6
president	•					
secretary		•				
treasurer			•			
board-member	•	•	•	•	•	
member	•	•	•	•	•	•

x_1 and x_2 are the intended referents

Attributes selected: 1) board-member (excluding x_6)

2) ¬treasurer (excluding x_3)

3) president ∨ secretary (excluding x_4 and x_5)

“a board-member, which is the president or the secretary, but not the treasurer”

instead of “the president and the secretary”

ATTACKING DEFICITS OF THE INCREMENTAL ALGORITHM – DEFICIT – AMBIGUITY (see Gardent 2002)

descriptors/objects	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
white	•	•	•	•	•	•	•	•	•	•	
dog		•						•	•	•	
cow			•	•	•	•	•				
big								•	•	•	
small						•	•				
medium-sized				•	•						
pitbul										•	
poodle									•		
holstein						•					
jersey					•						

ATTACKING DEFICITS OF THE INCREMENTAL ALGORITHM – DEFICIT – AMBIGUITY (cont'd)

x_5, x_6, x_9 and x_{10} are the intended referents

Attributes selected: 1) white (excluding x_{11}),

Then several possibilities, e.g.:

2) big ∨ cow (excluding x_1 and x_2)

3) Holstein ∨ ¬small (excluding x_7)

4) Jersey ∨ ¬medium (excluding x_4)

x_3 and x_8 still not excluded

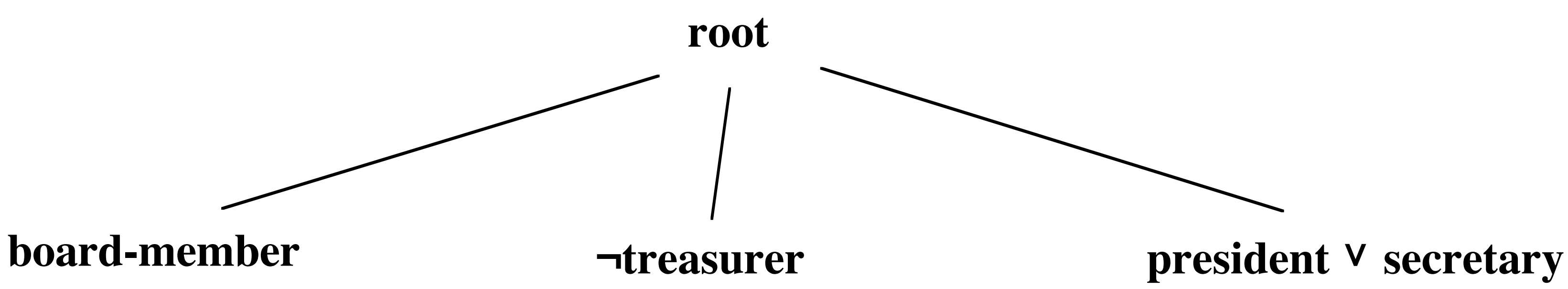
“the white things that are big or a cow, a Holstein or not small, and a Jersey or not medium” instead of

“the pitbul, the poodle, the Holstein, and the Jersey”

by Gardent's complete constraint-based search (1,4 sec)

COMPARISON WITH BEST-FIRST SEARCHING (I)

Example 1



11 msec

(7x faster than (Gardent 2002))

COMPARISON WITH BEST-FIRST SEARCHING (II)

Example 2

root

dog ∨ cow

pitbul ∨ poodle ∨ Jersey ∨ Holstein

dog ∨ Jersey ∨ Holstein

cow ∨ pitbul ∨ poodle

big ∨ medium-sized ∨ small

big ∨ medium-sized ∨ small

big ∨ medium-sized ∨ small

400 msec

(3,5x faster than (Gardent 2002))