Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 Second Semester 2017/2018

Lesson 12 23 March 2018

Planning as Search, Subgoals and Chunking

Planning as Search

The "paradigm" for planning is "Generate and Test".
Planning is the generation of a sequence of actions to transform i to a state $g \in \{G\}$
Planning requires search for a path through a graph of states.

Nils Nilsson defined a taxonomy of graph search algorithms includes the following
1) Breadth first search
2) Depth first search
3) Heuristic Search
4) Hierarchical Search

Nilsson unified Depth-First, Breadth First and Heuristic Search a single algorithm named GRAPHSEARCH.

The GRAPHSEARCH algorithm requires maintaining a list of "previously visited" states {C} (Closed list) and a list of available states to explore {O} (Open list).
As each state is tested, its neighbors are generated and added to the open list.
The state is then added to the closed list.

GRAPHSEARCH:

{O}←intial state
while {O} ≠ empty do
    Extract a new state s from {O}
    If $s \in \{G\}$ then HALT else add s to {C}
    Generate all neighbor states {N} of s.
    $\forall\ n \in \{N\}$ if $n \notin \{C\}$ then add n to open states {O}.

Breadth first, depth first and heuristic search are all variations on the same GRAPHSEARCH algorithm, depending on whether the Open list is a stack, queue or sorted.

1) Breadth first search - The Open list {O} is a Queue (First In, First Out)
2) Depth first search - The Open list {O} is a Stack (First In, Last Out)
3) Heuristic Search - The open list {O} is sorted by a Cost f(s) = g(s) + h(s)

Algorithmic Complexity of Search for Planning

We estimate algorithm complexity with the Order operator O().
Algorithm complexity order is equivalent for all linear functions.

$O(AN+B) = O(N)$

The algorithm complexity of graph search depends on

b: The branching factor; The average number of actions {A} possible in a state.
$b = E\{card(\{A\})\}$ (E{} is expectation)
d: Depth. The minimum number of actions from an initial state to a goal state.

Breadth First search: {O} is a queue (FIFO)

For breadth first search, finding the optimal path requires exhaustive search.
Computation Cost $O(b^d)$, memory $O(b^d)$.

This is a problem for humans, because human cognition has an important physiological limit: The size of short term working memory.

It is well known (and easily demonstrated by experiment) that human short-term working memory can hold, at most, 5+/- 2 concepts at one time.

The average person can remember 5 elements. Some people can hold 7. Some people can only hold 3.

To use breadth first planning a human must use a memory aid, such as a note pad.
Clearly human's use something more clever.

Depth First search: {O} is a stack (LIFO).

For depth first search, finding the optimal path requires exhaustive search, however
Computation Cost $O(b^d)$, memory O(d).

This can be done by humans for limited depth problems. ($d \leq 5$)
Depth first requires setting a maximum depth d_{max}.

However, both Breadth first and Depth search are exhaustive!
A cost of $O(b^d)$ is not acceptable for most real world problems.

Heuristic search: {O} is sorted based on the cost f(s) of a path through the state.

For Heuristic search, we reduce the order by reducing the branching factor:
This give computation and memory costs of $O(c^d)$ where $c \leq b$.

Heuristic Search is NOT exhaustive. The search avoids unnecessary branches.

For heuristic search, the cost of path through a state is f(s) = g(s) + h(s) where
Where g(s) is the cost of a path from the start state to the state s (easily calculated)
And h(s) is the cost of a path from the state to the goal state (unknown)

$$f(s) = g(s) + h(s)$$

Because h(s) is not known, it must be approximated with an estimate h*(s)

Optimality requires that both the cost function g(s) and the estimate, h*(s) meet the "optimality conditions".

Nilsson's Conditions for Optimal Search
Nilsson demonstrated the conditions under which a heuristic search algorithm is guaranteed to find the "optimal" plan.

Notation :
- i : initial state
- g : goal state $g \in \{G\}$
- $k(s_i, s_j)$: the minimal theoretical cost between states s_i and s_j
- $g^*(s) = k(i, s)$: The true cost of the shortest path from i to s.
- $h^*(s) = k(s, g)$: The true cost of the shortest path from s to $g \in \{G\}$.

$f^*(s) = g^*(s) + h^*(s)$ The true cost of the shortest path from i to g passing by s.

Problem: If we do not know the shortest path, how can we know h*(s)?
Solution estimate the costs.

Define:
- g(s) : estimated cost from i et s.
- h(s) : estimated cost from s to g
- f(s) = g(s) + h(s) estimated total cost of a path through s

Nilsson showed that whenever the estimated cost $f(s) \leq f^*(s)$,
the first path that is found from s_1 to s_2 will always be the shortest.
Thus g(s)=g*(s) because the first path from i to s has the least cost.

$f(s) \leq f^*(s)$ requires two conditions:

Condition 1: that the heuristic UNDER-ESTIMATES the cost.

$$h(s) \leq h^*(s)$$

Condition 2: that estimate h(s) is "monotonic". That is :

$$h(s_i) - h(s_j) \leq k(s_i, s_j)$$

This is almost always true whenever $h(s) \leq h^*(s)$!!

Nilsson called this the A* condition.
A* is "optimal" because the first path found is the shortest path.

Cost and Optimality of Heuristic Search

A key problem is defining heuristic search is the concept of numerical "cost" for executing an action.
Cost, $k(s_i, s_j)$, can be any numerical value, but must respect the optimality conditions for GRAPHSEARCH to be optimal.

Examples of cost: distance, time, Euros, risk, number of actions.

Whenever the cost metric is proportional to the length of the path, then Euclidean distance to the goal provides an "optimal" heuristic!

This is true for scalar multiples of distance, for example, time traveled or risk or cost or energy expended. (assuming constant speed, time = distance x 1/speed)

Note that for $h(s) = 0$, h(s) meets the optimality condition because $h(s) \leq h^*(s)$!!

This is dijkstra's algorithm, used for network routing.
We can speed up the search by using a better h(s) that 0! However, the first solution found is always the best solution.

Hierarchical Planning, Subgoals and Chunking

Cost of Search vs Optimality of Result

Note that there are TWO notions of cost!

1) Cost of executing the resulting plan.
2) Algorithmic complexity of the search (computational cost).

"Optimal" search means that the resulting plan is the cheapest possible plan.

For many real problem domains, the cost of search is MORE EXPENSIVE than the cost of executing the resulting plan.

In solving problems, humans sacrifice optimality to reduce the effort required for planning.

Subgoals

Subgoals can strongly reduce the algorithmic complexity search for a path through a network. A "Subgoal" is an abstract state that represents a set of states.

For example, to plan a route from campus to the train station.
1) Plan a route from Campus to the Quai d'Isere
2) Plan a route from the Quai d'Isere to the train station.

Define:

d : the minimal number for actions from the start state, i, to a goal state, g.
d_{is}: the minimal number for actions from the start state, i, to a subgoal s.
d_{sg}: the minimal number for actions from the subgoal s, to a goal state.

The subgoal divides the problem into two smaller subproblems: Plan a path from it to s, and plan a path from s to g.

$$O(b^{d_{is}} + b^{d_{sg}}) = O(b^{\max\{d_{is}, d_{sb}\}})$$

if $\max\{d_{is}, d_{sb}\} < d$, then the complexity is reduced.

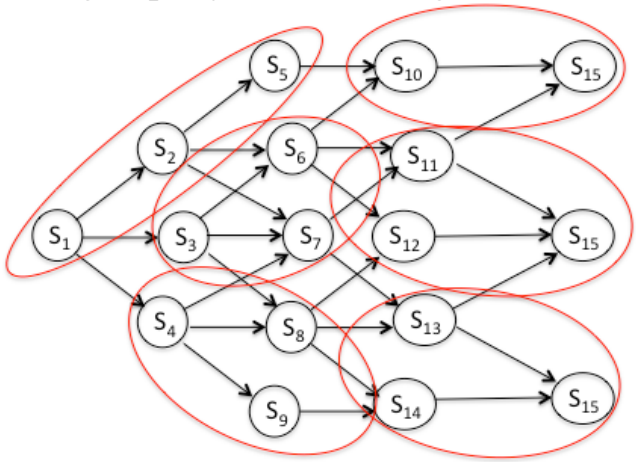
Hierarchical search sacrifices autonomy for scope.
Hierarchical search provides solutions to more complex problems, but there is no guarantee that the resulting solution is "optimal".

Planning as Search, Subgoals and Chunking

Two possible approaches to hierarchical planning are
1) Build a hierarchy of super-states
2) Define a hierarchy of operators. (meta-operators)

Hierarchy of states

An obvious approach is to group adjacent states together to form Super states.



Korf set this up as an optimisation problem and found that an "optimal" size for super-states was "e" (2.71828...)

Possible ways to group states

1) Group sets of states connected by a single action to a priviledges state
2) Define arbitrary connected sets of states
3) Group states by eliminative predicates.

A state is a conjunction of predicates $P_1() \wedge P_2() \wedge P_3() \wedge P_4()$

The state $P_1() \wedge P_2()$ represents the states $P_1() \wedge P_2() \wedge P_3()$ AND $P_1() \wedge P_2() \wedge \neg P_3()$

We can group states by deleting predicates and divide states into substates by adding predicates.

Sussman tried this with Blocks world and found that it was dependent on which states were eliminated. For example replacing

On(B,C) ∧ On(A,B) ∧ OT(A) with On(B,C) is not helpful.

Operators

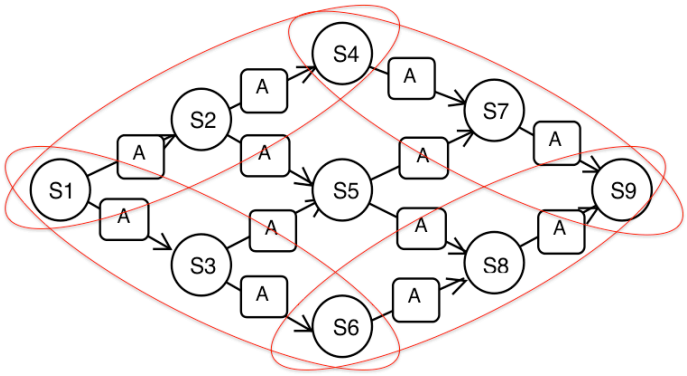
A more effective means is to group sequences of states and actions into "operators"

$S_1 - A_{12} \rightarrow S_2 - A_{23} \rightarrow S_3 - A_{34} \rightarrow S_4$

Is an operator to go from S_1 to S_4

The States translate to perceptual actions P() to verify the results of each action and to verify the pre-condition for the next action.

Operator $(S_1, S_4) = P(S_1) - A_{12} \rightarrow P(S_2) - A_{24} \rightarrow P(S_4)$



Human's do this to reduce the load on short term working memory. Simon called this "Chunking".

Operators are composed hierarchically, as needed, to accommodate the limits of human working memory.

Chunking

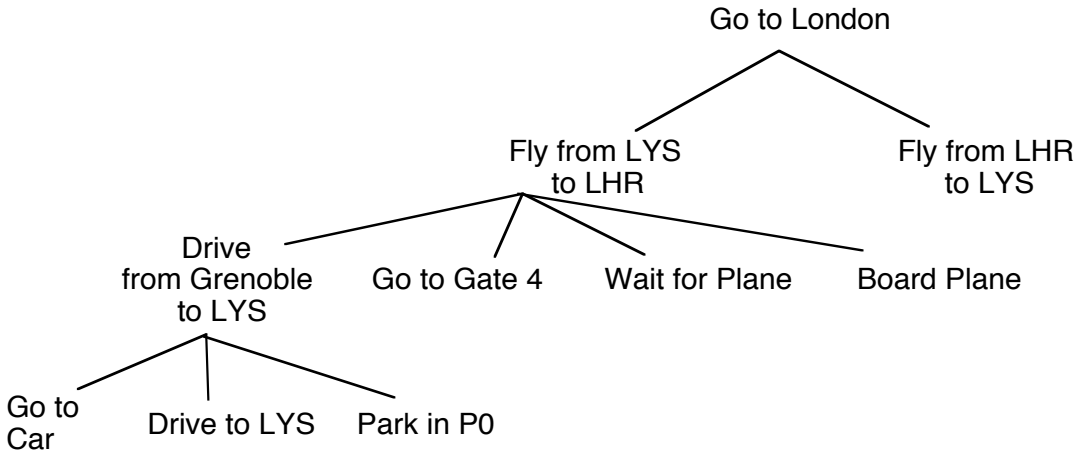
To speed up search and to overcome limits to short term memory, humans use a technique called "chunking".

Chunking is a process by which individual pieces of information are bound together into a meaningful whole. A chunk is a concept that represents a collection of concepts.

Chunking states into sets of states enables hierarchical planning.
Chunking can be applied hierarchically, with groups of states at each level represented by a single at the next level.

Example: Travel Planning

Consider the problem of planning a trip to Oxford.
First we choose whether to go by plane, train or bus.
Choosing plane, we select company and flights : Air France LYS-CDG-LHR
Then we plan the trip to Lyon Airport: Bus, Train or car
Then we plan the trip from Heathrow : Bus, Train or car
Then we plan the trip to the train station to catch the bus to LYS, etc.



Spatial organization of knowledge

Navigation requires a "map". The classic map for path planning is a "network of places".

A place is defined as
1) A name
2) An inclusion test (typically a predicate at(x))
3) A list of "adjacent" places that can be reached by a single action.

The set of places compose a network. (The network of places).

Navigation planning requires finding a sequence of places that lead from one place to another. The search for a path generates a tree of possible paths.

In addition, the network of places also serves as a memory structure. Humans associate experiences and actions with each place.

Consider for example the tramway map of Grenoble:

Each stop is a Place, defined by a name (E.g. Gabriel Fauré) and a list of adjacent places (Berlioz, Les Taillées, Biblioteques). Travel from each tram stop to the next is a Primitive Action. Once the tram doors close, you go directly to the next stop, whether you want to or not.

Each Place is associated with Perceptual Memories, and possible actions.

Knowing the tramway system can mean many things.

Knowing the network of places allows you to navigate the system (Go from any station to any other station).

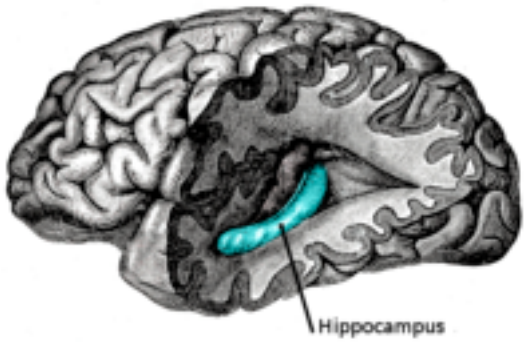
However, when may associate other kinds of knowledge with each stop:

- Images (of things that can be seen at the stop) (visual memory)
- Sounds: that can be heard at the stop (auditory memory)
- Words: statements that you have heard or made about the stop (Propositions)
- Experiences that happened to you at the stop (Episodic Memory)
- Functions: things you can do at the stop (Operational memory)

A network of places provides a structure for organizing experience and cognitive abilities.

The hippocampus

Navigation is a fundamental ability for survival. In mammals, navigation memory is provided by a network structure located in the hippocampus.



From Wikipedia: The hippocampus belongs to the limbic system and plays important roles in the consolidation of information from short-term memory to long-term memory, and in spatial memory that enables navigation. The hippocampus is located under the cerebral cortex and in primates in the medial temporal lobe.

The hippocampus has been studied extensively as part of a brain system responsible for spatial memory and navigation. Many neurons in the rat and mouse hippocampus respond as place cells: that is, they fire bursts of action potentials when the animal passes through a specific part of its environment. Hippocampal place cells interact extensively with head direction cells, whose activity acts as an inertial compass, and conjecturally with grid cells in the neighboring ento-rhinal cortex.

A place is a region is a region of space that is connected to other places. Places are typically associated with several kinds of knowledge (cognitive ability).