What is (Single-Agent) Search?

- A problem solving method (my "hammer")
  - there are a few other problem solving methods around, but
  - I have no idea what they are good for :)
- Problem → Graph Representation → Search (Graph Traversal)→ Path → Solution
- Search can look for a (satisfying) answer or the best (optimizing) answer.

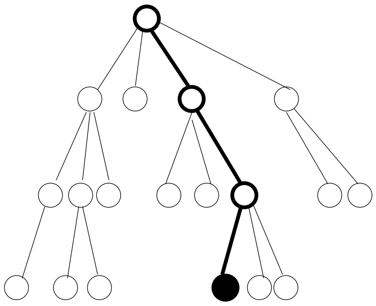
Graph Representation

Node: A state of our problem world.

Edge: A (legal) transition between two states.

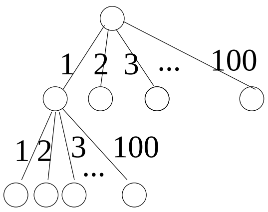
Root: The initial problem setting.

Goal: A special node (or path) in the graph representing a solution.



Example I - Simple Number Partitioning

For a given number C, find two numbers A and B, such that $A * B = C$ and $A + B$ is minimal, and $A, B, C \in \{1..100\}$.



Simple Number Partitioning - Comments

1. Two nested loops do the trick:

```
for (A=1; A<=100; A++) {
  for (B=1; A<=100; B++) {
    if (A*B == C && A*B < S) {
      _A = A; _B = B; S = A*B;
    }
  }
}
```

2. This is really dumb, but it works!
3. We can get smart later.
4. Some issues: completeness, correctness, efficiency (representation/knowledge)...
5. Loops work here, because this problem is regular and simple. Just think chess or Rubik's Cube, or...

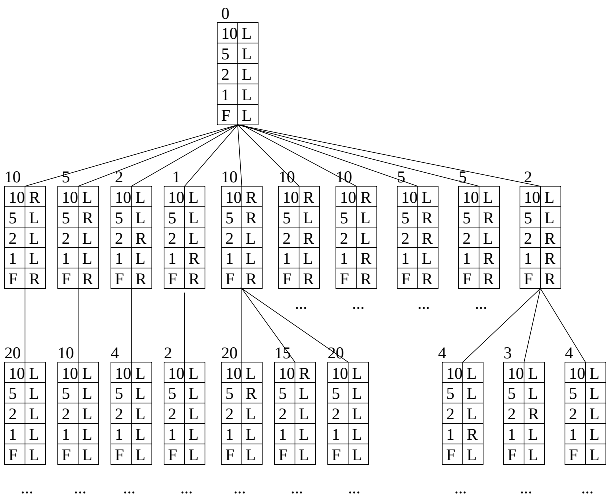
Example II - The Rolling Stones Problem

- The 4 Rolling Stones need to get to a concert which starts in 17 minutes on the other side of a gorge.
- There is a bridge, crossing it, they need 10, 5, 2, and 1 minute each.
- At most two people can cross the bridge at a time at the speed of the slower, but need the one and only flashlight available.
- Can they make it in time (17 minutes)?
- Yes or No - what is the fasted time they can make it (proof)?

The Rolling Stones Problem - Graph Representation

- A **state** is defined by:
  1. Who is on which side (left or right),
  2. where is the flashlight (left or right),
  3. what time was used so far.
- A **state transition** (move) is the flashlight crossing the bridge from its current side taking one or two people with it.
- The **cost** of a transition is the time of the crossing.
- The **start** is all 4 and flashlight are on the left side of the bridge.
- The **goal** is that all 4 are on the bridge right side.

The Rolling Stones Problem - Partial Search Space



The Rolling Stones Problem - Some Considerations

- How many moves are there in the start node/position?
- Does it make sense to move one guy from left to right?
- Does it make sense to consider "double" moves from right to left?
- Are you convinced of the correctness of the solution?
- How can we prove that we found the shortest time?
- What if you need a year to compute the answer?

How to "Search" a Search Space

- uninformed traversal (no knowledge)

  depth-first: First try to expand children, then siblings.

  breadth-first: First expand siblings (same levels) then "go down" to next lower level.

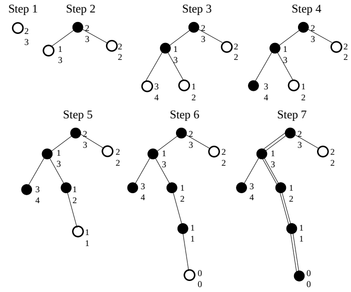
- informed traversal (needs knowledge)

  best-first: Expand the nodes "closest" to the goals. Heuristic knowledge is used to estimate "closeness".

  A*: Expands the node(s) expected to be on the shortes path to the goal.

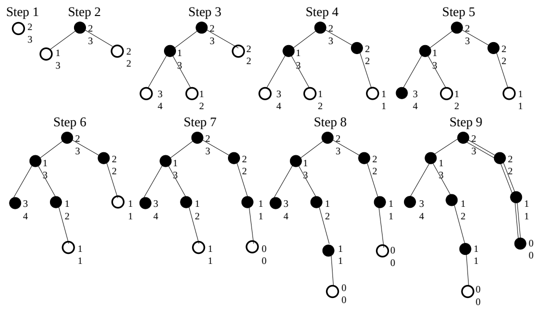
  IDA*: Like A*, but depth-first with a limit on the maximal path-length permitted (which will be increased).

Depth-First Search




```
DepthFirst( StartState )
{
  Push( OpenStack, StartState );
  Success = FALSE;
  DO {
    CurrentState = Pop(OpenStack);
    IF( Solution( CurrentState ) )
      Success = TRUE;
    ELSE
      FOREACH( Child( CurrentState ) ) DO
        Push( OpenStack, Child( CurrentState ) );
  } UNTIL( Success OR Empty( OpenStack ) );
  IF( Success ) RETURN( CurrentState );
  ELSE RETURN( NULL );
}
```


Breadth-First Search




```
BreadthFirst( StartState )
{
  Store( OpenFIFO, StartState );
  Success = FALSE;
  DO {
    CurrentState = GetFirstIn( OpenFIFO );
    IF( Solution( CurrentState ) )
      Success = TRUE;
    ELSE
      FOREACH( Child( CurrentState ) ) DO
        Store( OpenFIFO, Child( CurrentState ) );
  } UNTIL( Success OR Empty( OpenFIFO ) );
  IF( Success ) RETURN( CurrentState );
  ELSE RETURN( NULL );
}
```


(Effective) Search-Space Size

- Clever representation of the search space can make huge difference.

- But there are limits to clever representations, eventually, we have to deal with LARGE search spaces.

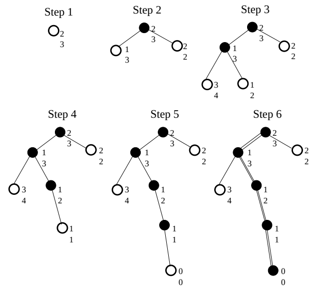
- Brute force (complete enumeration) even with all the speed in the world won't do it.

- The only hope then is to use magic...

Knowledge is Magic

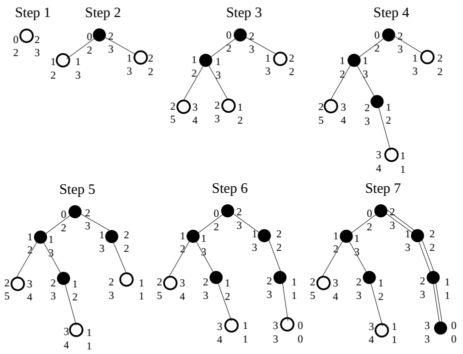
- Between "no" and "absolute" knowledge lies the gray zone of heuristic knowledge.
- The better the knowledge, the smaller the effective search space.
- Heuristic knowledge can (usually) be abstracted into an estimate of the distance (cost) from any node in the search space to the nearest goal node.

Best-First Search



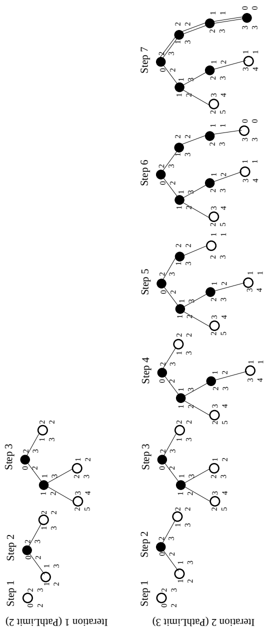

```
BestFirst( StartState )
{
  Insert( OpenSortedList, StartState );
  Success = FALSE;
  DO {
    CurrentState = GetBest( OpenSortedList );
    IF( Solution( CurrentState ) )
      Success = TRUE;
    ELSE
      FOREACH( Child( CurrentState ) ) DO
        InsertSortedOnH( OpenSortedList,
              Child( CurrentState ) );
  } UNTIL( Success OR Empty( OpenSortedList ) );
  IF( Success ) RETURN( CurrentState );
  ELSE RETURN( NULL );
}
```


A*




```
A_STAR( StartState )
{
  StartState.g = 0;
  Insert( OpenSortedList, StartState );
  Success = FALSE;
  DO {
    CurrentState = GetBest( OpenSortedList );
    IF( Solution( CurrentState ) )
      Success = TRUE;
    ELSE
    {
      FOREACH( Child( CurrentState ) ) DO {
        IF( IsIn( OpenSortedList( Child( CurrentState ) ) ) )
        {
          OldState = Get( OpenSortedList( Child( CurrentState ) ) );
          OldState.g = min( OldState.g, Child( CurrentState ).g );
        }
        ELSIF( IsIn( ClosedList( Child( CurrentState ) ) ) )
          PropagateG( Get( ClosedList( Child( CurrentState ) ) ) );
        ELSE
          InsertSortedOnF( OpenSortedList, Child( CurrentState ) );
      }
      Insert( ClosedList, CurrentState );
    }
  } UNTIL( Success OR Empty( OpenSortedList ) );
  IF( Success ) RETURN( CurrentState );
  ELSE RETURN( NULL );
}
```

IDA*



```
IDA_STAR( StartState )
{
  PathLimit = H( StartState ) - 1;
  Success = FALSE;
  DO {
    PathLimit ++;
    StartState.g = 0;
    Push( OpenStack, StartState );
    DO {
      CurrentState = Pop( OpenStack );
      IF( Solution( CurrentState ) )
        Success = TRUE;
      ELSIF( PathLimit >= ( CurrentState.g
             + H( CurrentState ) ) )
        FOREACH( Child( CurrentState ) ) DO
          Push( OpenStack, Child( CurrentState ) );
    } UNTIL( Success OR Empty( OpenStack ) );
  } UNTIL( Success OR ResourceLimitsReached() );
  IF( Success ) RETURN( CurrentState );
  ELSE RETURN( NULL );
}
```


The Choice of Algorithm

!!!! IS EASY !!!!

Depends on:

- availability and quality of knowledge
- goal location
- goal density
- search space properties

Magic (Knowledge) is Hard

!This is the most important slide of this talk!
!This is the most important slide of this talk!
!This is the most important slide of this talk!

1. Knowledge acquisition is hard.
2. Knowledge verification is hard.
3. Knowledge representation is hard.
4. Knowledge tuning is hard.
5. Knowledge evaluation is hard.
6. Ignoring knowledge is easy :)

Heuristics - I

- If we can traverse the entire search space (tic-tac-toe, rolling-stone problem,...) no knowledge is required.
- If we had perfect knowledge, we had no problem - no search would be required.
- Perfect knowledge would just mean to have a correct relative ordering among all states in the search space (total or partial order). The "exact" value of states is not needed.
- Heuristic knowledge covers the middle ground: search space is too large to search exhaustively and we do not have perfect, but only "guiding" knowledge.
- Heuristics are "rules of thumb" - they can be wrong at times.

Heuristics - II

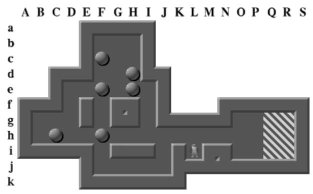
Where does knowledge come from?

- Simplify domain rules such that in the resulting problem the estimation of the distance to goal can be calculated using a (cheap) function.
- E.g.: Time for all on the left to cross the bridge is at least the time needed by the slowest person.
- Properties of knowledge:
  Admissibility - Never overestimate the distance to goal.
- Only admissible heuristics allow for guaranteed optimal solutions (best-first, A*, IDA*).

Heuristics - III

- Currently mainly humans create that knowledge.
- Expensive, cumbersome, error-prone, **boring**,...
- How to use computers to create it is still not (fully) understood.
- Machine Learning might eventually lead to useful tools - but it is still not there, especially compared against knowledge generated by humans.
- Recently some advances, but that is only a sorry beginning...

Example III - Sokoban



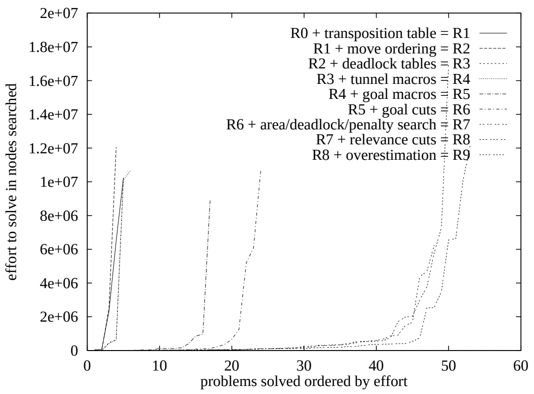
- invented in the early 80s in Japan
- push stones to goals
- push one stone at a time
- only push stones
- http://xsokoban.lcs.mit.edu/xsokoban.html
- test set of 90 (hard!?) problems
- up to 35 stones with 4 directions to go: large branching factor
- solutions from 81 to 672 pushes long
- around 10^{98} possible states in a 20x20 maze

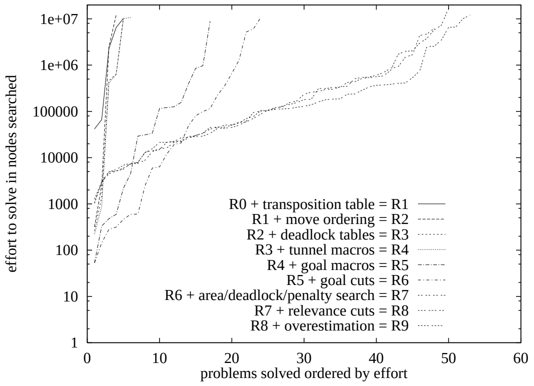
Sokoban - Representation, Search Space Properties and Knowledge

- stone-goal matching → minimum-cost bipartide matching
- likely successful moves → move ordering
- transpositions → transposition table
- deadlocks → pattern databases and pattern searches
- move chains → macro moves
- ...

Sokoban - Benefits of Knowledge

Technique added	# Solved
IDA*	0
Lower Bound	0
Trans. Table	5
Move Ordering	4
Deadlock Tables	5
Tunnel Macros	6
Goal Macros	17
Goal Cuts	24
Pattern Search	48
Relevance Cuts	50
Overestimation	54
RRR	57





Single-Agent vs. Adversary Search

- "Backup Rule" says what the parents node value is, given child node values.
- In Single-Agent Search this is usually easy: Max or Min.
- In some other domains, this might be different: Minimax, ProbiMax, Multiplication, ...
- ... but anything said here today applies to those domains as well, because the backup rule really does not matter much when your main problems are efficiency, speed, and knowledge.

Current Problems and The Future

- On-line knowledge gathering (problem restructuring).
- Stepping outside to observe and change problem solving behavior.
- ...

Some (Philosophical) Concluding Remarks about AI - !!Rant Alert!!

- AI is chasing something impossible to achieve.
- Therefore, AI will never be successful - don't go into it, if you want to be successful.
- But "chasing" has brought a few "techniques" that proved useful to solve "real" problems (Deep Blue :)
- The problem with AI starts with the "I" and does not stop with the "A": What is intelligence? What is even artificial?
- My definition of intelligence:
  An entity that can go about its task in a

  changing environment with (expected) increasing efficiency.
- I don't believe it is important how that is achieved: Neither the "hardware" nor the "software" define intelligence, only the resulting behavior with respect to the task at hand and the environment.
- external vs. internal definition of intelligence
- "The appearance of intelligence is its definition."
- Search is one of many tools to achieve the illusion of intelligence, because currently there is very little adaptive behavior in the search technology - it might never be.
- ...