AND/OR graphs

- Some problems are best represented as achieving subgoals, some of which achieved simultaneously and independently (AND)

- Up to now, only dealt with OR options

[Diagram of AND/OR graph with nodes: Possess TV set, Steal TV, Earn Money, Buy TV]
Searching AND/OR graphs

- A solution in an AND-OR tree is a *sub tree* whose *leaves* are included in the goal set.

- Cost function: sum of costs in AND node
  \[ f(n) = f(n_1) + f(n_2) + \ldots + f(n_k) \]

- How can we extend A* to search AND/OR trees? The AO* algorithm.
AND/OR search

- We must examine several nodes simultaneously when choosing the next move
AND/OR Best-First-Search

- Traverse the graph (from the initial node) following the best current path.
- Pick one of the unexpanded nodes on that path and expand it. Add its successors to the graph and compute $f$ for each of them.
- Change the expanded node’s $f$ value to reflect its successors. Propagate the change up the graph.
- Reconsider the current best solution and repeat until a solution is found.
AND/OR Best-First-Search example

1. A (5)

2. A
   B (3) (9) C (4) D (5)

3. A
   B (3) (9) C (4) D (10)
   E (10) F (4)
AND/OR Best-First-Search example
A Longer path may be better

A

B

C

D

G

H

E

F

I

J

Unsolvable

B

C

D

A

G

H

E

F

I

J

Unsolvable

Unsolvable
Interacting Sub goals

A

D

C (5)

E (2)
AO* algorithm

1. Let $G$ be a graph with only starting node $INIT$.
2. Repeat the followings until $INIT$ is labeled SOLVED or $h(INIT) > FUTILITY$
   a) Select an unexpanded node from the most promising path from $INIT$ (call it NODE)
   b) Generate successors of NODE. If there are none, set $h(NODE) = FUTILITY$ (i.e., NODE is unsolvable); otherwise for each SUCCESSOR that is not an ancestor of NODE do the following:
      i. Add SUCCESSSOR to G.
      ii. If SUCCESSOR is a terminal node, label it SOLVED and set $h(SUCCESSOR) = 0$.
      iii. If SUCCESSPRR is not a terminal node, compute its h
c) Propagate the newly discovered information up the graph by doing the following: let $S$ be set of SOLVED nodes or nodes whose $h$ values have been changed and need to have values propagated back to their parents. Initialize $S$ to Node. Until $S$ is empty repeat the followings:

i. Remove a node from $S$ and call it CURRENT.

ii. Compute the cost of each of the arcs emerging from CURRENT. Assign minimum cost of its successors as its $h$.

iii. Mark the best path out of CURRENT by marking the arc that had the minimum cost in step ii.

iv. Mark CURRENT as SOLVED if all of the nodes connected to it through new labeled arc have been labeled SOLVED.

v. If CURRENT has been labeled SOLVED or its cost was just changed, propagate its new cost back up through the graph. So add all of the ancestors of CURRENT to $S$. 
An Example
An Example

(8) A
An Example

(1) B
(2) C

A

[12] 4
5

D

(8)

[13] 5
An Example

A

B

C

D

4

5

2

4

5

2

15

13

8

4

2

2
An Example
An Example

[15]

Solved

Solved

Solved
Real Time A*

- Considers the cost (> 0) for switching from one branch to another in the search
- Example: path finding in real life
Another Example

Current State = S
- $f(A) = 3 + 5 = 8$
- $f(B) = 2 + 4 = 6$

Current State = B
- $f(S) = 2 + 8 = 10$
- $f(A) = 4 + 5 = 9$
- $f(C) = 1 + 5 = 6$
- $f(E) = 4 + 2 = 6$

Current State = C
- $f(H) = 2 + 4 = 6$
- $f(B) = 1 + 6 = 7$
Current State = H
\[ f(C) = 2 + 7 = 9 \]

Current State = C
\[ f(B) = 1 + 6 = 7 \]
\[ f(H) = \infty \]

Current State = B
\[ f(S) = 2 + 8 = 10 \]
\[ f(A) = 4 + 5 = 9 \]
\[ f(E) = 4 + 2 = 6 \]
\[ f(C) = \infty \]

Current State = E
\[ f(B) = 4 + 9 = 13 \]
\[ f(D) = 3 + 2 = 5 \]
\[ f(F) = 1 + 1 = 2 \]
Another Example

Current State = F
f(E) = 1 + 5 = 6

Current State = E
f(D) = 3 + 2 = 5
f(B) = 4 + 9 = 13
f(F) = ∞

Current State = D
f(G) = 2 + 0 = 2
f(E) = 3 + 13 = 16

Visited Nodes = S, B, C, H, C, B, E, F, E, D, G

Path = S, B, E, D, G