# Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings


Blai Bonet
Dept. de Computación
Universidad Simón Bolívar
Caracas, Venezuela

Héctor Geffner
Dept. de Tecnología
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

# Motivation

- **Heuristic search** methods can be **efficient** but lack common foundation: IDA*, AO*, Alpha-Beta, . . .
- **Dynamic programming** methods such as Value Iteration are **general** but not as efficient:
  - Single algorithm for wide range of models: Det, MDPs, Games, AND/OR, . . .
  - yet VI is exhaustive
- This work aims to bring these two types of methods together to obtain:
  - efficiency, generality, and understanding!

# Result

- A simple algorithm, **Learning Depth-First Search** (LDFS), capable of solving a **wide range** of deterministic and non-deterministic models; based on three ideas
  - Depth-First Search
  - Lower bounds
  - Learning
- For some models, LDFS **reduces to state-of-the-art algorithms:**
  - **Deterministic Models:** LDFS = IDA* w/ transposition tables
  - **Game Trees:** (Bounded) LDFS = Alpha-Beta w/ null windows (MTD) [Plaat et. al, 1996]
  - On others, like AND/OR and MDPs, LDFS yields **new algorithms**

# Basic Intuitions: IDA*

**IDA***

- Performs iterative Depth-First Searches with certain *bound*
- Prune action $a$ in node $n$ leading to node $n'$ when

$$g(n) + c(a, n) + h(n') > bound$$

# Basic Intuitions: IDA*

**IDA***

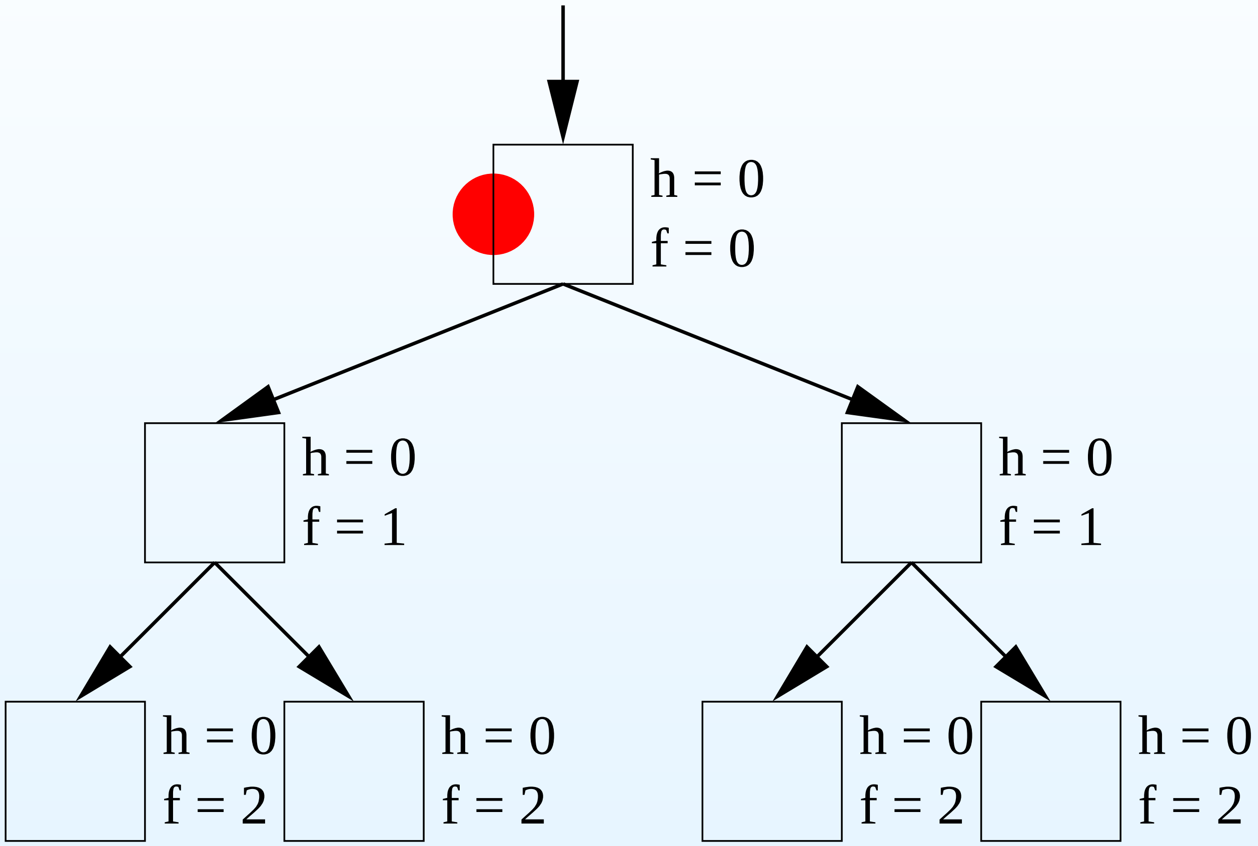
- Performs iterative Depth-First Searches with certain $bound$
- Prune action $a$ in node $n$ leading to node $n'$ when

$$g(n) + c(a,n) + h(n') > bound$$

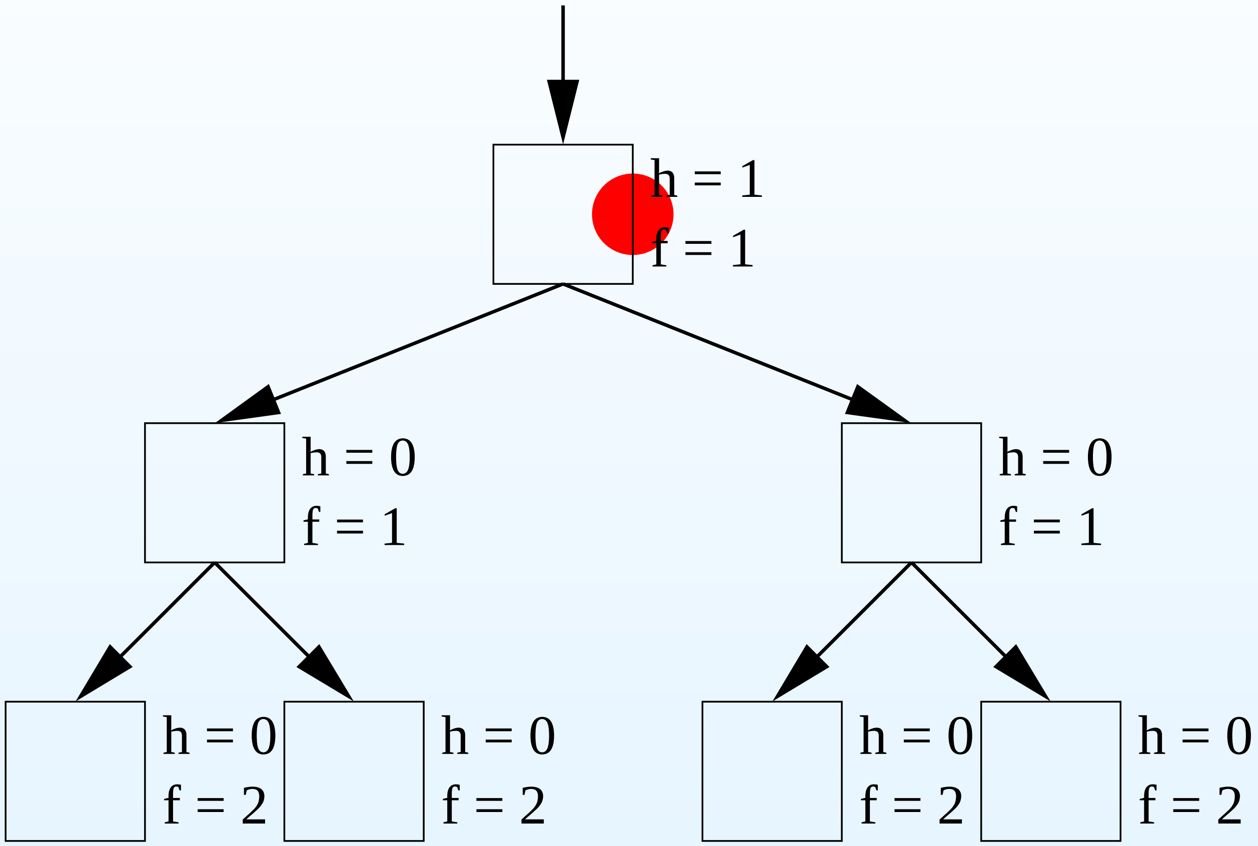
**IDA* w/ Transposition Table (Cost Revisions [Reinefeld & Marsland, 1994])**

- As IDA*: performs iteratives DFS's with certain $bound$
- Upon backtracking from node $n$, **revise** heuristic value $h(n)$ (stored in TT) to the new lower bound

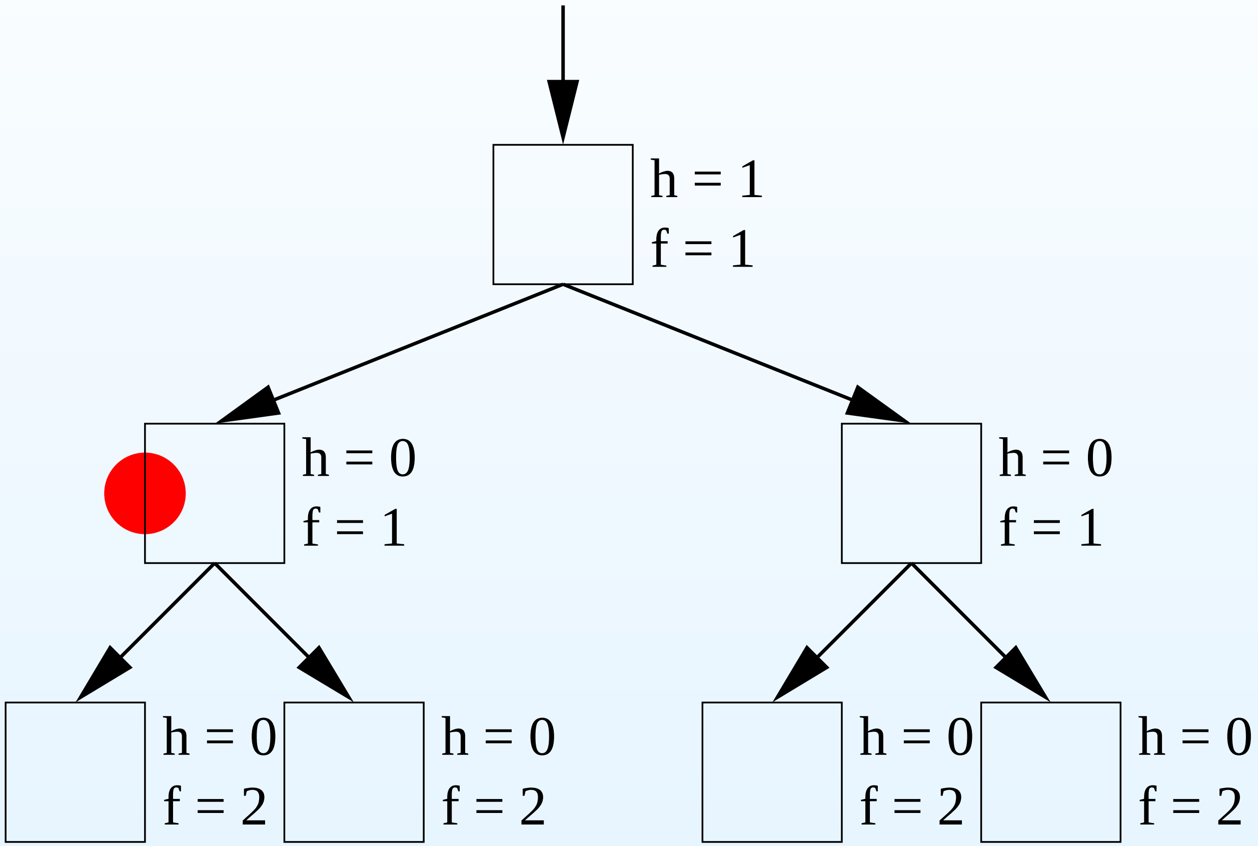
# Ilustration: IDA* w/ Transposition Table



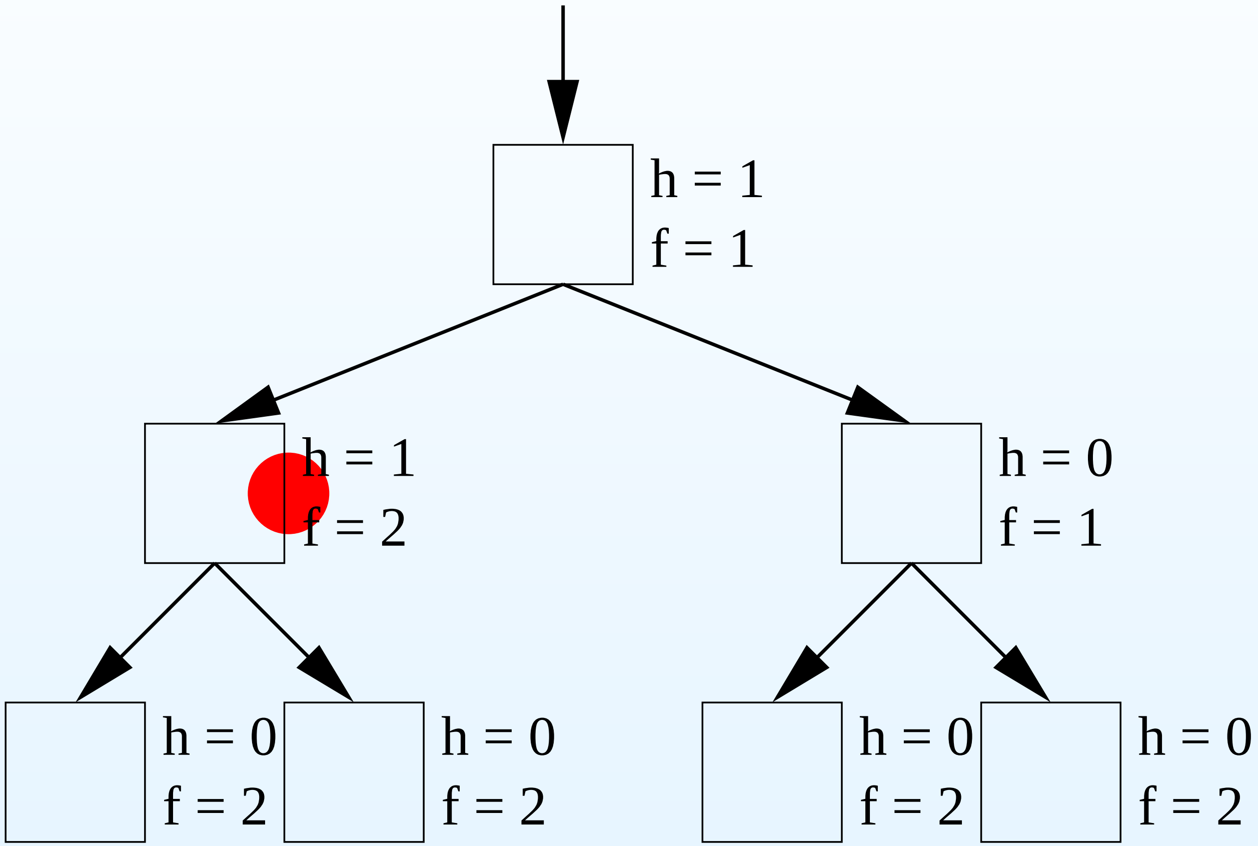
# Ilustration: IDA* w/ Transposition Table



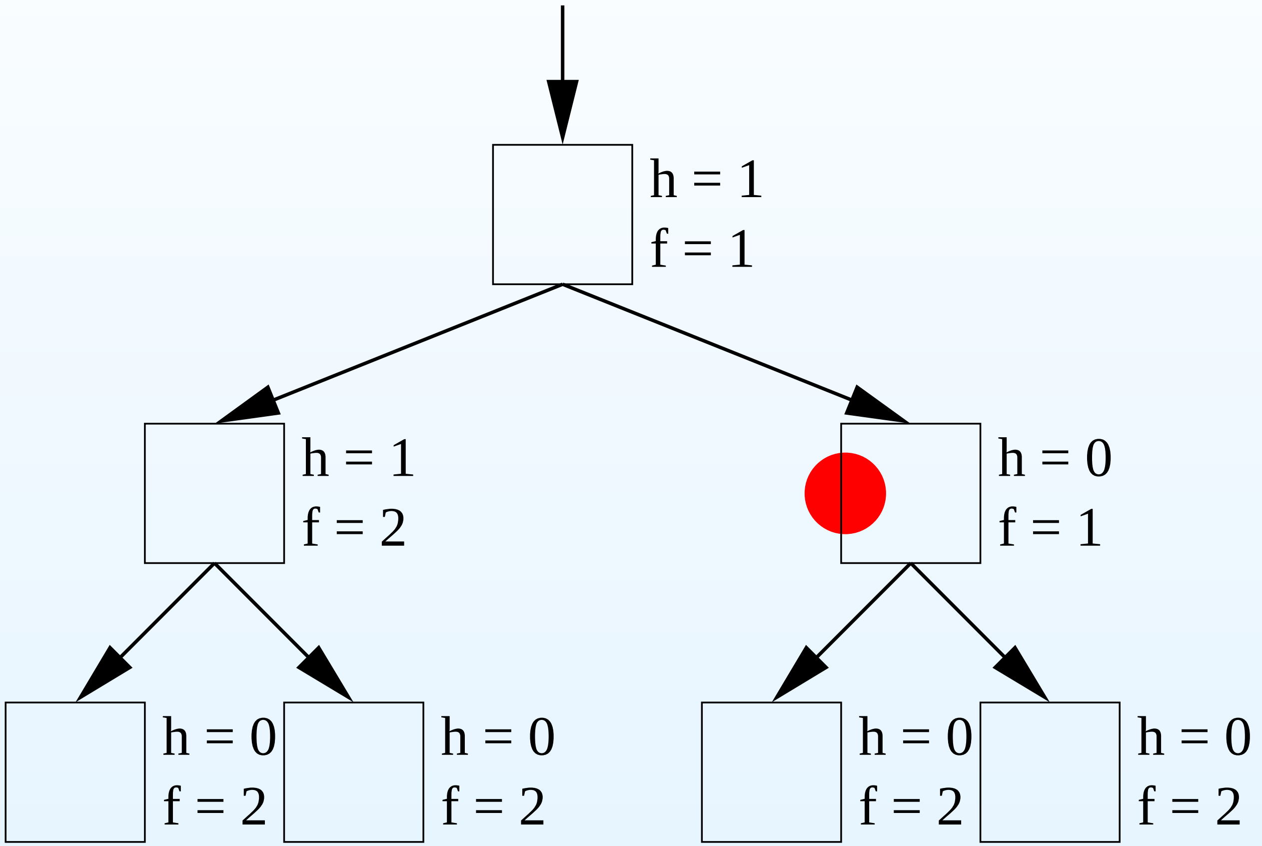
# Ilustration: IDA* w/ Transposition Table



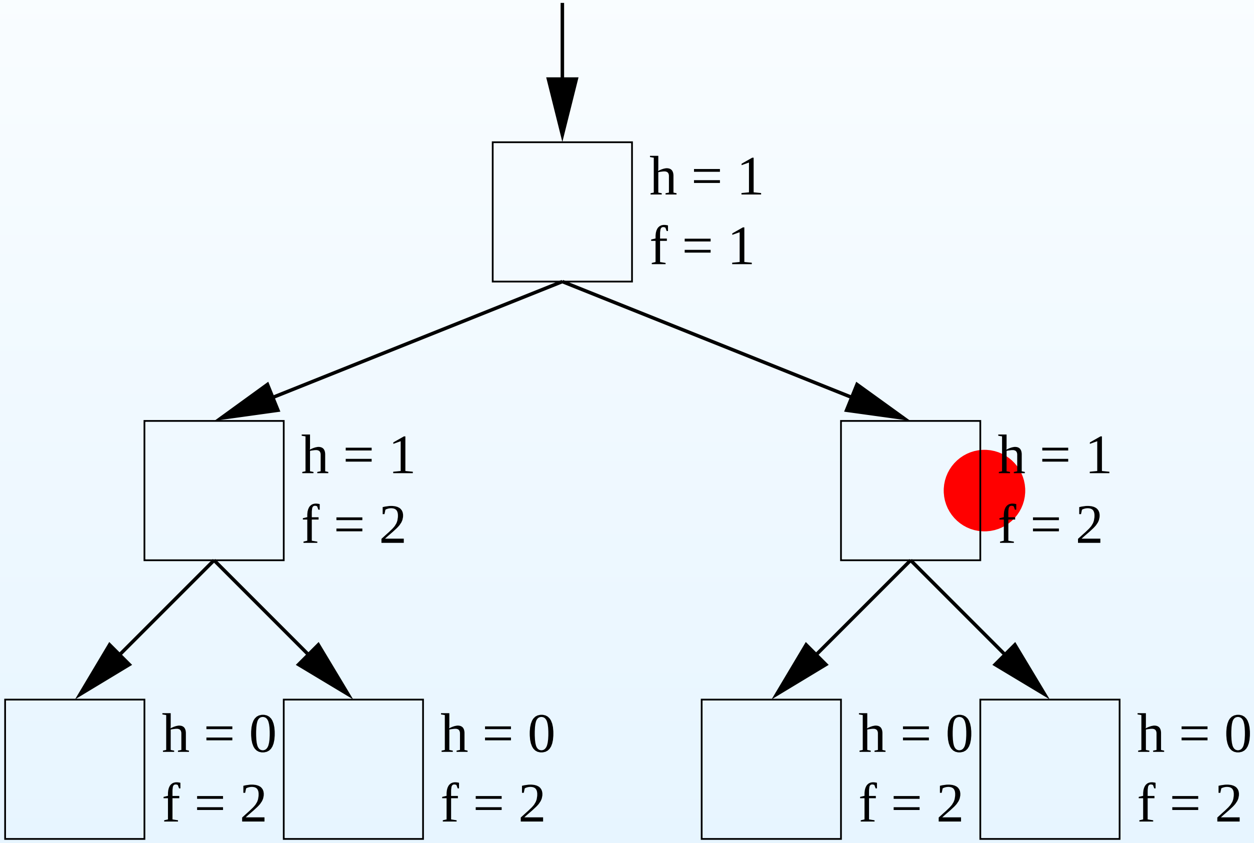
# Ilustration: IDA* w/ Transposition Table



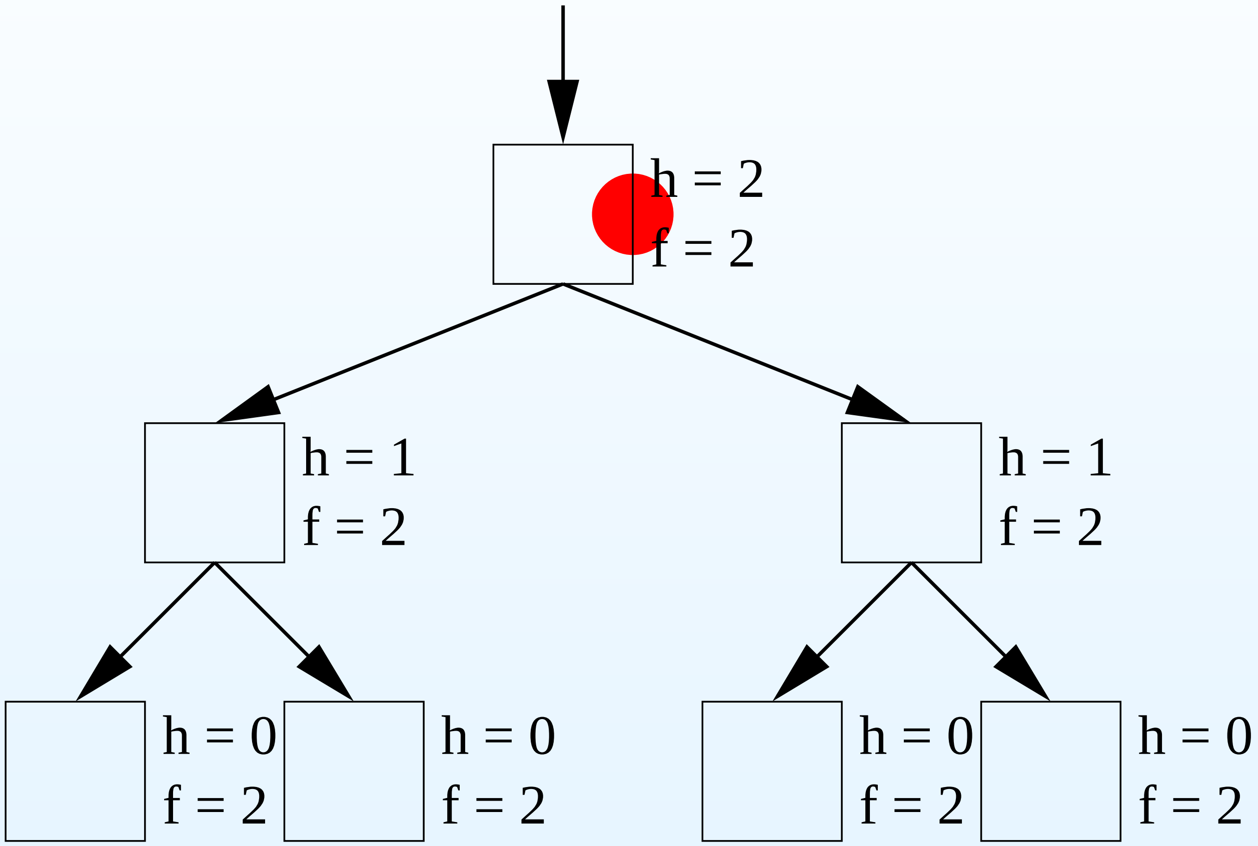
# Ilustration: IDA* w/ Transposition Table



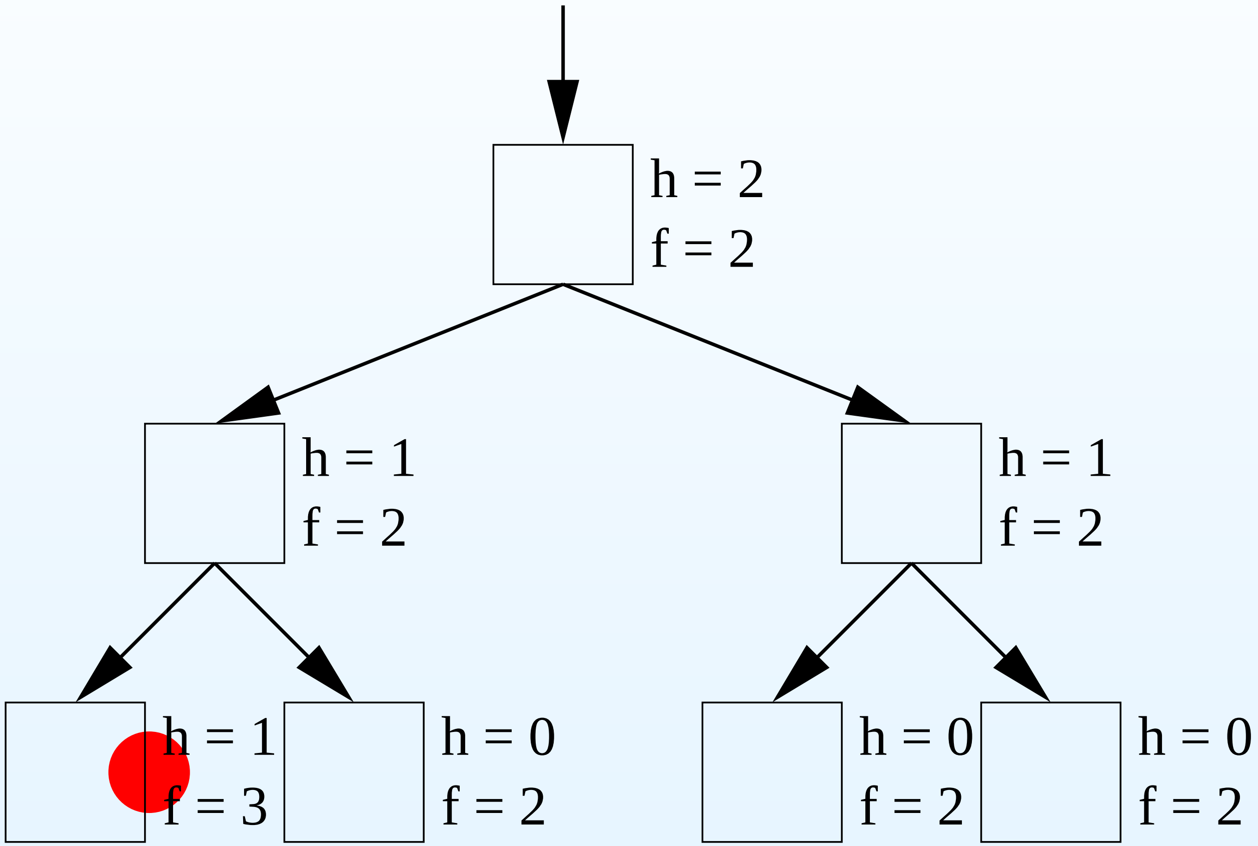
# Ilustration: IDA* w/ Transposition Table



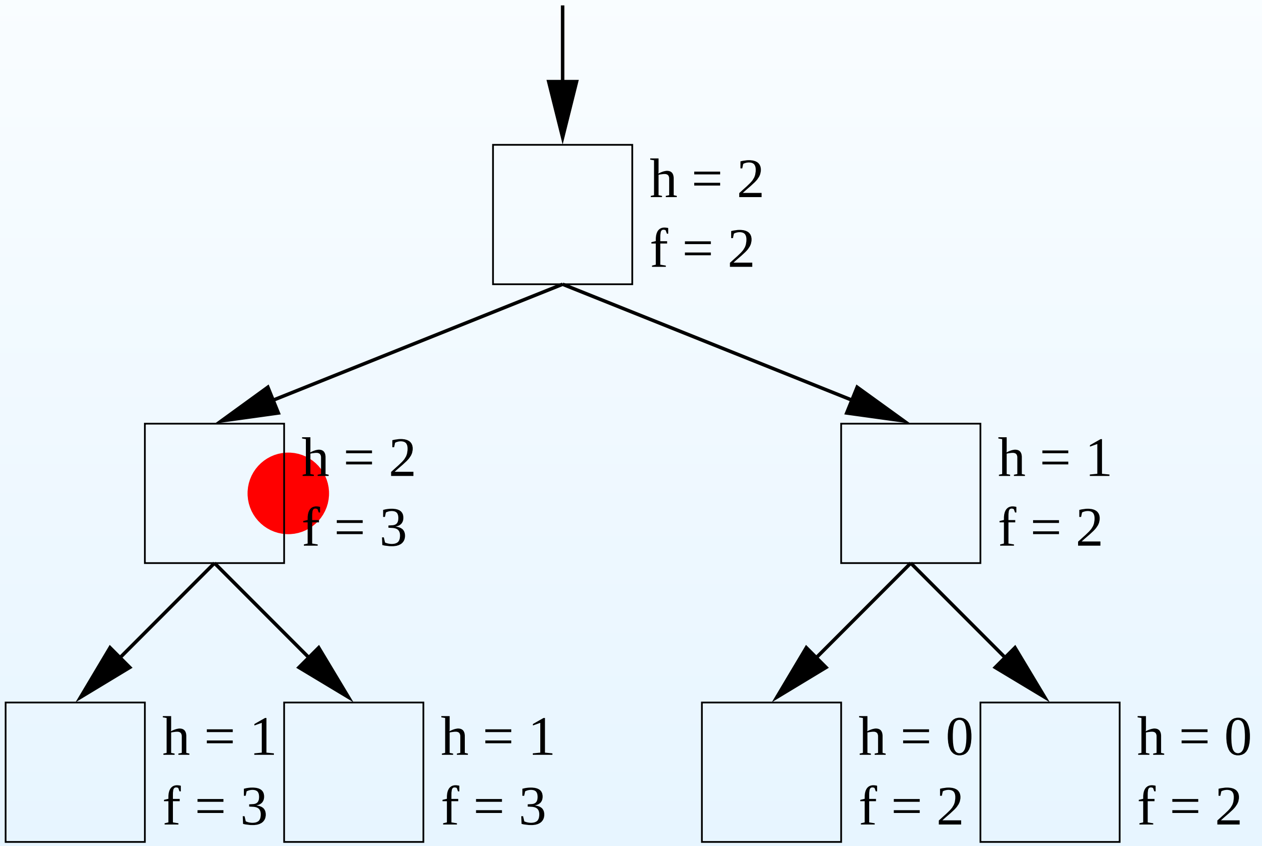
# Ilustration: IDA* w/ Transposition Table



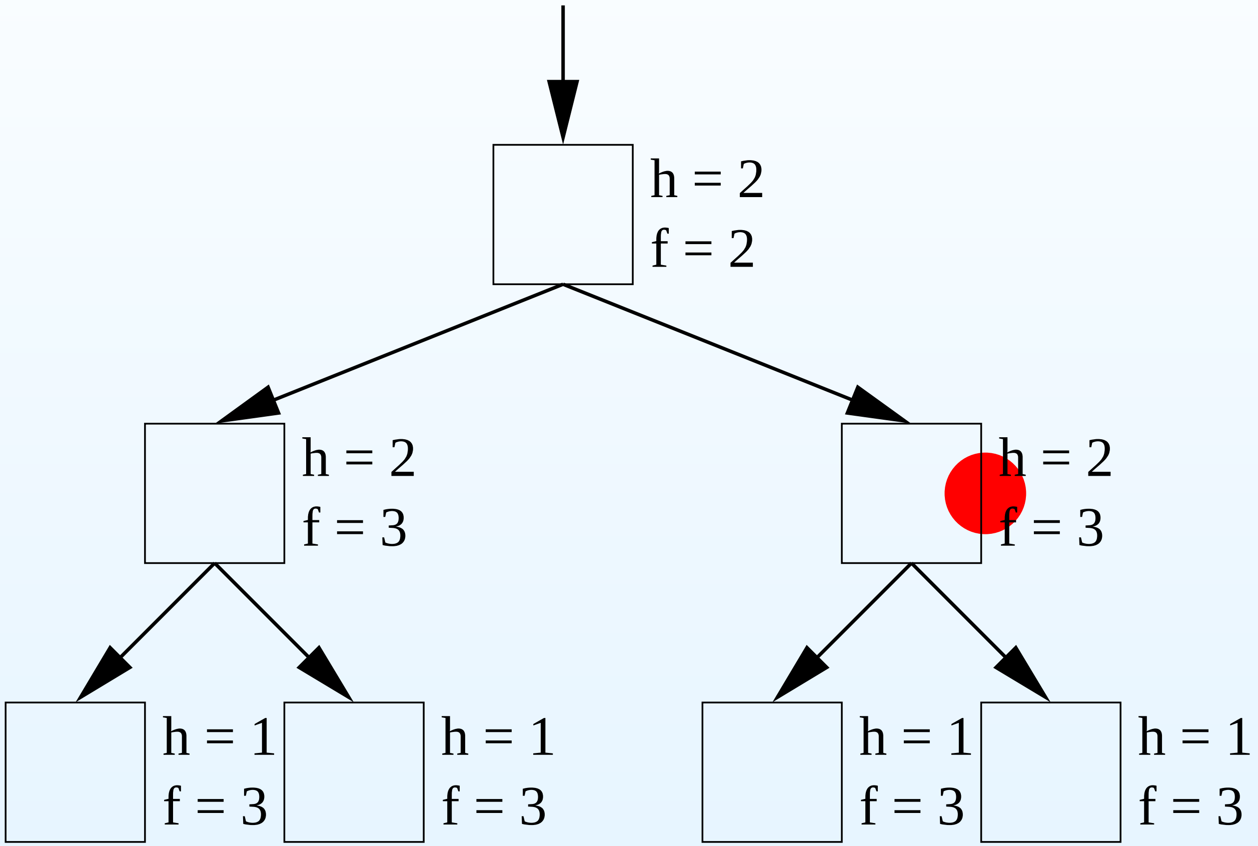
# Ilustration: IDA* w/ Transposition Table



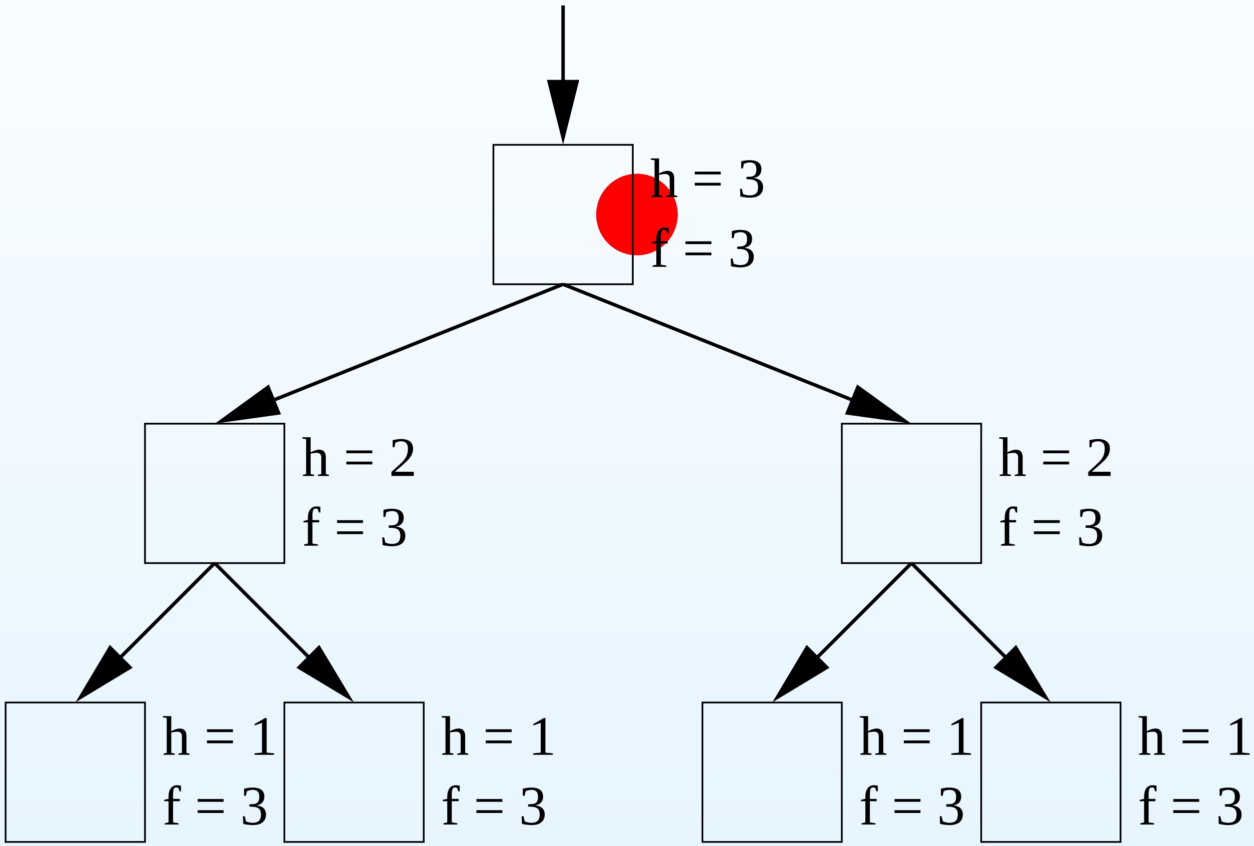
# Ilustration: IDA* w/ Transposition Table



# Ilustration: IDA* w/ Transposition Table



# Ilustration: IDA* w/ Transposition Table



# Monotone Heuristics and IDA*

- Monotonicity is a property of $h$ that says

$$f(n) \;\leq\; f(n')$$

  for each node $n$ and succesor $n'$; i.e. **non-decreasing $f$-values along paths**

- With monotone $h$, IDA* has **important properties:**
  - The bound for each iteration equals $h(n_0)$
  - An iteration either finds solution or increases $h(n_0)$ which is next bound
  - The revision of the heuristic (TT) is just

$$h(n) \;:=\; \min_{a \in A(n)} c(n,a) + h(n')$$

- Path $(n_0, a_0, n_1, \ldots, a_i, n_{i+1})$ is **transversed** by IDA* (w/ TT and Monotone $h$) iff

$$bound \;=\; f(n_0) \;\leq\; f(n_1) \;\leq\; f(n_2) \;\leq\; \ldots \;\leq\; f(n_i) \;\leq\; bound$$

$$f(n_0) \;=\; f(n_1) \;=\; f(n_2) \;=\; \ldots \;=\; f(n_i)$$

# IDA* + TT and Monotone $h$: Reformulated (Generalized)

- If $Q_h(a,n) = c(a,n) + h(n')$, the algorithm can be expressed as iterations that:
  - Starting from $n_0$, perform DFS along actions $a$ such that
$$h(n) \;=\; Q_h(a,n)$$
  - Backtrack at tip nodes $n$ (i.e. with no such $a$'s), restoring consistency of $h(n)$:
$$h(n) \;:=\; min_{a \in A(n)} \, Q_h(a,n)$$

# IDA* + TT and Monotone $h$: Reformulated (Generalized)

- If $Q_h(a,n) = c(a,n) + h(n')$, the algorithm can be expressed as iterations that:
  - Starting from $n_0$, perform DFS along actions $a$ such that

$$h(n) \;=\; Q_h(a,n)$$

  - Backtrack at tip nodes $n$ (i.e. with no such $a$'s), restoring consistency of $h(n)$:

$$h(n) \;:=\; min_{a \in A(n)}\, Q_h(a,n)$$

- **Good News:** reformulation is very **general**; other models can solved be **efficiently** by suitable choice of $Q_h(a,n)$:

| | |
|---|---|
| $Q_h(a,n) = c(a,n) + \max_{n'} h(n')$ | for MAX AND/OR GRAPHS |
| $Q_h(a,n) = c(a,n) + \sum_{n'} P_a(n' \vert n) h(n')$ | for MDPS |
| $Q_h(a,n) = \max_{n'} h(n')$ | for GAME TREES |
| ... | ... |

## IDA* + TT and Monotone $h$: Reformulated (Generalized)

- If $Q_h(a,n) = c(a,n) + h(n')$, the algorithm can be expressed as iterations that:
  - Starting from $n_0$, perform DFS along actions $a$ such that

$$h(n) \; = \; Q_h(a,n)$$

  - Backtrack at tip nodes $n$ (i.e. with no such $a$'s), restoring consistency of $h(n)$:

$$h(n) \; := \; min_{a \in A(n)} \, Q_h(a,n)$$

- **Good News:** reformulation is very **general**; other models can solved be **efficiently** by suitable choice of $Q_h(a,n)$:

| | |
|---|---|
| $Q_h(a,n) = c(a,n) + \max_{n'} h(n')$ | for MAX AND/OR GRAPHS |
| $Q_h(a,n) = c(a,n) + \sum_{n'} P_a(n' \vert n) h(n')$ | for MDPS |
| $Q_h(a,n) = \max_{n'} h(n')$ | for GAME TREES |
| $\ldots$ | $\ldots$ |

- We call this algorithm **LDFS** for **Learning Depth-First Search**

# LDFS: The Code

```
LDFS-DRIVER(n_0)
begin
    repeat solved := LDFS(n_0) until solved
    return (V, π)
end

LDFS(n)
begin
    if n is SOLVED or terminal then
        if n is terminal then V(n) := c_T(n)
        Mark n as solved return true
    flag := false                                        % EXPANSION
    foreach a ∈ A(n) do
        if Q_V(a,n) > V(n) then continue
        flag := true
        foreach n' ∈ F(a,n) do
            flag := LDFS(s') & flag             % Recursion
            if ¬flag then break
        if flag then break
    if flag then
        π(n) := a
        Mark n as SOLVED                                 % LABELING
    else
        V(n) := min_{a∈A(n)} Q_V(a,n)                    % UPDATE
    return flag
end
```

# Rest of the Talk: Outline

- What are Q-factors and where they come from?
- Why LDFS works?
- Properties and relation to other algorithms
- Extensions and empirical results for MDPs

# A bit of Background: Models

- a discrete and finite states space $S$,
- an initial state $s_0 \in S$,
- a non-empty set of terminal states $S_T \subseteq S$,
- actions $A(s) \subseteq A$ applicable in each non-terminal state,
- a function that maps states and actions into *sets* of states $F(a,s) \subseteq S$,
- action costs $c(a,s)$ for non-terminal states $s$, and
- terminal costs $c_T(s)$ for terminal states.

DETERMINISTIC: $|F(a,s)| = 1$ (OR Graphs),

NON-DETERMINISTIC: $|F(a,s)| \geq 1$ (AND/OR graphs),

MDPS: probabilities $P_a(s'|s)$ for $s' \in F(s,a)$ that add up to 1

...

# Solutions

- (Optimal) Solutions can be expressed in terms of value function $V$ satisfying **Bellman** equation:

$$V(s) = \begin{cases} c_T(s) & \text{if } s \text{ is terminal} \\ \min_{a \in A(s)} Q_V(a,s) & \text{otherwise} \end{cases}$$

where $Q_V(a,s)$ stands for the cost-to-go value defined as:

$$Q_V(a,s) = c(a,s) + V(s'),\ s' \in F(a,s) \quad \text{for OR GRAPHS}$$

$$Q_V(a,s) = c(a,s) + \max_{s' \in F(a,s)} V(s') \quad \text{for MAX AND/OR GRAPHS}$$

$$Q_V(a,s) = c(a,s) + \sum_{s' \in F(a,s)} V(s') \quad \text{for ADD AND/OR GRAPHS}$$

$$Q_V(a,s) = c(a,s) + \sum_{s' \in F(a,s)} P_a(s'|s) V(s') \quad \text{for MDPS}$$

$$Q_V(a,s) = \max_{s' \in F(a,s)} V(s') \quad \text{for GAME TREES}$$

# Solutions

- (Optimal) Solutions can be expressed in terms of value function $V$ satisfying **Bellman** equation:

$$V(s) = \begin{cases} c_T(s) & \text{if } s \text{ is terminal} \\ \min_{a \in A(s)} Q_V(a,s) & \text{otherwise} \end{cases}$$

where $Q_V(a,s)$ stands for the cost-to-go value defined as:

$$\begin{array}{ll} Q_V(a,s) = c(a,s) + V(s'),\ s' \in F(a,s) & \text{for OR GRAPHS} \\ Q_V(a,s) = c(a,s) + \max_{s' \in F(a,s)} V(s') & \text{for MAX AND/OR GRAPHS} \\ Q_V(a,s) = c(a,s) + \sum_{s' \in F(a,s)} V(s') & \text{for ADD AND/OR GRAPHS} \\ Q_V(a,s) = c(a,s) + \sum_{s' \in F(a,s)} P_a(s'|s) V(s') & \text{for MDPS} \\ Q_V(a,s) = \max_{s' \in F(a,s)} V(s') & \text{for GAME TREES} \end{array}$$

- An **optimal policy** can be recovered from the solution of Bellman equation as:

$$\pi(s) = \operatorname{argmin}_{a \in A(s)} Q_V(a,s)$$

## LDFS: Learning Depth-First Search

- Assuming monotone and admissible value function $V$ (i.e. $h$):
    - Start from $n_0$ and perform DFS along actions such that

$$V(n) = Q_V(a,n)$$

    - Backtrack when there is no such action and update $V(n)$ to

$$V(n) := min_{a \in A(n)} Q_V(a,n)$$

- LDFS solves all models above, **except MDPs with cyclic solutions**
- LDFS is **equivalent** to IDA* w/ TT on Deterministic models (OR graphs)

# Value Iteration Algorithm

- There is an algorithm that is almost as general, and even simpler: **Value Iteration**
- Value Iteration **doesn't search, just makes updates:**
  - Iterate until convergence:

$$\text{For all node } n \text{ do:} \quad V(n) := min_{a \in A(n)} Q_V(a,n)$$

- VI is pretty good when **all states fit in memory** (e.g. around $10^6$ states)

# Why is VI less general and less effective than LDFS?

- It doesn't work in the presence of dead-ends (states with $V(s) = \infty$)

# Why is VI less general and less effective than LDFS?

- It doesn't work in the presence of dead-ends (states with $V(s) = \infty$)
- It doesn't exploit LBs/Heuristic information; **it is an exhaustive method**

## Why is VI less general and less effective than LDFS?

- It doesn't work in the presence of dead-ends (states with $V(s) = \infty$)
- It doesn't exploit LBs/Heuristic information; **it is an exhaustive method**
- For example, if $Q_V(a,n) > V^*(n)$, LDFS will never consider action $a$ at $n$;
- E.g. IDA* never explores child $n'$ of $n$ if $Q_h(a,n) > h(n)$ and $h$ is monotone

## Find-and-Revise: An Abstraction that Searchs and Updates

- Find-and-Revise is a theoretical model for analysis; defined in terms of
  - Greedy Graph $G_V$: contains nodes $n$ reachable from $n_0$ by applying **greedy** actions $a$; i.e. those with $Q_V(a,n) = min_a\, Q_V(a,n)$
  - Consistent nodes: those such that $V(n) = min_a\, Q_V(a,n)$

# Find-and-Revise: An Abstraction that Searchs and Updates

- Find-and-Revise is a theoretical model for analysis; defined in terms of
  - Greedy Graph $G_V$: contains nodes $n$ reachable from $n_0$ by applying **greedy** actions $a$; i.e. those with $Q_V(a,n) = min_a\, Q_V(a,n)$
  - Consistent nodes: those such that $V(n) = min_a\, Q_V(a,n)$

- Find-and-Revise iterates:
  - Find an inconsistent node $n$ in greedy graph $G_V$
  - Update (revise) $V$ at node $n$

  until no such state exists

# Find-and-Revise: An Abstraction that Searchs and Updates

- Find-and-Revise is a theoretical model for analysis; defined in terms of
  - Greedy Graph $G_V$: contains nodes $n$ reachable from $n_0$ by applying **greedy** actions $a$; i.e. those with $Q_V(a,n) = min_a\ Q_V(a,n)$
  - Consistent nodes: those such that $V(n) = min_a\ Q_V(a,n)$

- Find-and-Revise iterates:
  - Find an inconsistent node $n$ in greedy graph $G_V$
  - Update (revise) $V$ at node $n$

  until no such state exists

- **THM:** Find-and-Revise solves all models if initial $V$ is admissible and monotone

# LDFS is an instance of Find-and-Revise!

LDFS is a Find-and-Revise that:

- Finds with a DFS search that backtracks upon inconsistent states
- Upon backtracking updates inconsistent states and ancestors
- Keeps track of SOLVED states to avoid re-exploration (**labeling**)

# Some Properties of LDFS

**Additive Models (e.g. OR graphs, Additive AND/OR, ...)**

- Each iteration of LDFS either increases the value of $s_0$ or labels $s_0$ as solved
- Hence, number of iterations bounded by $V^*(s_0) - h(s_0)$
- Does not deal with probabilistic models with cyclic solutions (MDPs)

# Some Properties of LDFS

**Additive Models (e.g. OR graphs, Additive AND/OR, ...)**

- Each iteration of LDFS either increases the value of $s_0$ or labels $s_0$ as solved
- Hence, number of iterations bounded by $V^*(s_0) - h(s_0)$
- Does not deal with probabilistic models with cyclic solutions (MDPs)

**Max Models (e.g. Max AND/OR, Game Trees, ...)**

- An iteration of LDFS may no increase the value of $s_0$ neither label it
- Yet a simple variation, called **Bounded LDFS**, restores such property
- **Bounded LDFS = Alpha-Beta w/ null windows** (MTD) [Plaat et. al, 1996]

# Variations Needed for Cyclic MDPs

- Cannot do the bottom-up labeling because of cycles

# Variations Needed for Cyclic MDPs

- Cannot do the bottom-up labeling because of cycles
- In MDPs, the value function converges to $V^*$ **asymptotically**, thus need to replace consistency

$$min_a \, Q_V(a,n) \, - \, V(n) \, = \, 0$$

with $\varepsilon$-consistency:

$$min_a \, Q_V(a,n) \, - \, V(n) \, \leq \, \varepsilon$$

# Variations Needed for Cyclic MDPs

- Cannot do the bottom-up labeling because of cycles
- In MDPs, the value function converges to $V^*$ **asymptotically**, thus need to replace consistency

  $$min_a \; Q_V(a,n) - V(n) = 0$$

  with $\varepsilon$-consistency:

  $$min_a \; Q_V(a,n) - V(n) \leq \varepsilon$$

- IDA* might perform an exponential number of iterations in problems with real costs, and so do LDFS in MDPs

# Variations Needed for Cyclic MDPs

- Cannot do the bottom-up labeling because of cycles
- In MDPs, the value function converges to $V^*$ **asymptotically**, thus need to replace consistency
  $$min_a \; Q_V(a,n) \; - \; V(n) \; = \; 0$$
  with ε-consistency:
  $$min_a \; Q_V(a,n) \; - \; V(n) \; \leq \; \varepsilon$$
- IDA* might perform an exponential number of iterations in problems with real costs, and so do LDFS in MDPs
- The resulting algorithm is called LDFS+

# A Bit of Empirical Results

- Four domains: noisy 8-puzzle, racetracks, rooms, tree
- Algorithms: VI, LRTDP, ILAO, HDP, LDFS+
- Two (monotone) heuristics: zero, min-min relaxation

| algorithm | small | big | bigger | ring-1 | ring-2 | ring-3 | ring-4 | ring-5 | ring-6 |
|---|---|---|---|---|---|---|---|---|---|
| $\|S\|$ | 9,394 | 22,532 | 51,941 | 429 | 1,301 | 5,949 | 33,243 | 94,396 | 352,150 |
| $V^*(s_0)$ | 14.459 | 26.134 | 50.570 | 7.498 | 10.636 | 13.093 | 18.530 | 24.949 | 31.142 |
| $h_{min\text{-}min}(s_0)$ | 11 | 18 | 37 | 6 | 9 | 11 | 15 | 20 | 25 |
| VI$(h_{min\text{-}min})$ | 1.080 | 3.824 | 14.761 | 0.022 | 0.105 | 0.611 | 5.198 | 23.168 | 197.964 |
| LRTDP$(h_{min\text{-}min})$ | 0.369 | 3.169 | 12.492 | 0.006 | 0.027 | 0.138 | 2.173 | 15.361 | 243.130 |
| ILAO$(h_{min\text{-}min})$ | 0.813 | 4.739 | 20.190 | 0.008 | 0.034 | 0.463 | 11.428 | 37.598 | — |
| HDP$(h_{min\text{-}min})$ | 0.468 | 5.357 | 30.174 | 0.007 | 0.034 | 0.180 | 2.159 | 11.473 | 153.150 |
| LDFS+$(h_{min\text{-}min})$ | **0.196** | **1.077** | **4.542** | **0.003** | **0.014** | **0.083** | **1.022** | **4.892** | **80.068** |
| VI$(h=0)$ | 1.501 | 5.289 | 21.701 | 0.027 | 0.124 | 0.774 | 7.281 | 34.501 | 354.917 |
| LRTDP$(h=0)$ | 0.880 | 6.232 | 29.836 | **0.012** | 0.109 | 0.356 | 6.005 | 171.829 | — |
| ILAO$(h=0)$ | 2.430 | 14.200 | 54.208 | 0.024 | 0.109 | 0.908 | 11.863 | 71.103 | — |
| HDP$(h=0)$ | 2.440 | 30.955 | 174.698 | 0.032 | 0.149 | 0.927 | 11.957 | 96.398 | — |
| LDFS+$(h=0)$ | **0.792** | **3.417** | **16.080** | 0.013 | **0.057** | **0.353** | **4.390** | **24.732** | **310.019** |

# Summary

- A simple algorithm, **Learning Depth-First Search** (LDFS), capable of solving a **wide range** of deterministic and non-deterministic models; based on three ideas
  - Depth-First Search
  - Lower bounds
  - Learning
- For some models, LDFS **reduces to state-of-the-art algorithms:**
  - **Deterministic Models:** LDFS = IDA* w/ transposition tables
  - **Game Trees:** (Bounded) LDFS = Alpha-Beta w/ null windows (MTD) [Plaat et. al, 1996]
  - On others, like AND/OR and MDPs, LDFS yields **new algorithms**
- Competitive results for LDFS+ on **Markov Decision Processes**