An Algorithm better than AO*?

Blai Bonet
Universidad Simón Bolívar
Caracas, Venezuela

Héctor Geffner
ICREA and Universitat Pompeu Fabra
Barcelona, Spain

7/2005

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
- **Question:** can we the get the best of both; i.e., **generality** and **efficiency**?
- **Answer** is **yes**, combining their key ideas:

  Admissible Heuristics (Lower Bounds)
  Learning (Value Updates as in LRTA*, RTDP, etc)

What does proposed integration give us?

An **algorithm schema**, called **LDFS**, that is **simple**, **general**, and **efficient**:

- **simple** because it can be expressed in a few lines of code; indeed

$$\textbf{LDFS = Depth First Search} + \textbf{Learning}$$

- **general** because it handles many models: OR Graphs (IDA*), AND/OR Graphs (AO*), Game Trees (Alpha-Beta), MDPs, etc.

- **efficient** because it reduces to state-of-the-art algorithms in many of these models, while in others, yields new competitive algorithms; e.g.

$$\textbf{LDFS} = \begin{cases} \textbf{IDA* + TT} & \textbf{for OR-Graphs} \\ \textbf{MTD}(-\infty) & \textbf{for Game Trees} \end{cases}$$

We also show that **LDFS** better than **AO*** over **Max AND/OR Graphs** . . .

What does proposed integration give us? (cont'd)

- Like LRTA*, RTDP, and LAO*, **LDFS** combines **lower bounds** with **learning**, but motivation and goals are slightly different

- By accounting for and generalizing **existing algorithms**, we aim to uncover the **three key computational ideas** that underlie them all so that **nothing else is left out**. These ideas are:

  Depth First Search
  Lower Bounds
  Learning

- It is also useful to know that, say, new **MDP** algorithm, reduces to well-known and tested algorithms when applied **OR-Graphs** or **Game Trees**

Models

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

- $\textsc{Deterministic}$: $|F(a,s)| = 1$,
- $\textsc{Non-Deterministic}$: $|F(a,s)| \geq 1$,
- $\textsc{MDPs}$: probabilities $P_a(s'|s)$ for $s' \in F(s,a)$ that add up to 1 . . .

Solutions

(Optimal) Solutions can all 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:

$$\begin{array}{ll} c(a,s) + V(s'),\; s' \in F(a,s) & \text{for OR Graphs} \\ c(a,s) + \max_{s'\in F(a,s)} V(s') & \text{for Max AND/OR Graphs} \\ c(a,s) + \sum_{s'\in F(a,s)} V(s') & \text{for Add AND/OR Graphs} \\ c(a,s) + \sum_{s'\in F(a,s)} P_a(s'|s)V(s') & \text{for MDPs} \\ \max_{s'\in F(a,s)} V(s') & \text{for Game Trees} \end{array}$$

A **policy** (solution) π maps states into actions, must be **closed around** s_0, and is **optimal** if $\pi(s) = \text{argmin}_{a\in A(s)} Q_V(a,s)$ for V satisfying **Bellman**

Value Iteration (VI): A general solution method

1. Start with **arbitrary** cost function V
2. Repeat until residual over all s is 0 (i.e., LHS $=$ RHS)

   Update $V(s) := \min_{a \in A(s)} Q_V(a, s)$ for **all** s
3. Return $\pi_V(s) = \text{argmin}_{a \in A(s)} Q_V(a, s)$

- VI is **simple** and **general** (models encoded in form of Q_V), but also **exhaustive** (considers all states) and affected by **dead-ends** ($V^*(s) = \infty$)
- Both problems solvable using **initial state** s_0 and **lower bound** V . . .

Find-and-Revise: Selective VI Schema

Assume V **admissible** ($V \leq V^*$) and **monotonic** ($V(s) \leq \min_{a \in A(s)} Q_V(a, s)$)
Define s **inconsistent** if $V(s) < \min_{a \in A(s)} Q_V(a, s)$)

1. `Start with a` **lower bound** V
2. `Repeat until no more states found in` a.
   a. **Find inconsistent** s **reachable** `from` s_0 `and` π_V
   b. **Update** $V(s)$ `to` $\min_{a \in A(s)} Q_V(a, s)$
3. `Return` $\pi_V(s) = \text{argmin}_{a \in A(s)} Q_V(a, s)$

- Find-and-Revise yields optimal π in at most $\sum_s V^*(s) - V(s)$ iterations (provided integer costs and no probabilities)
- Proposed **LDFS** = **Find-and-Revise** with:
  - **Find** = **DFS that backtracks on inconsistent states** that
  - **Updates** states on backtracks, and
  - **Labels** as **Solved** states s with no inconsistencies beneath

Learning in Depth-First Search (LDFS)

```
LDFS-DRIVER(s_0)
begin
    repeat solved := LDFS(s_0) until solved
    return (V, π)
end

LDFS(s)
begin
    if s is solved or terminal then
        if s is terminal then V(s) := c_T(s)
        Mark s as SOLVED
        return true

    flag := false
    foreach a ∈ A(s) do
        if Q_V(a, s) > V(s) then continue
        flag := true
        foreach s' ∈ F(a, s) do
            flag := LDFS(s') & [Q_V(a, s) ≤ V(s)]
            if ¬flag then break
        if flag then break

    if flag then
        π(s) := a
        Mark s as SOLVED
    else
        V(s) := min_{a∈A(s)} Q_V(a, s)

    return flag
end
```

Properties of LDFS and Bounded LDFS

LDFS computes π^* for **all models** if V admissible (i.e. $V \leq V^*$)

- For **OR-Graphs** and **monotone** V,

$$\text{LDFS} \quad = \quad \text{IDA}^* + \text{TRANSPOSITION TABLES}$$

- For **Game Trees** and $V = -\infty$,

$$\text{BOUNDED LDFS} \quad = \quad \text{MTD}(-\infty)$$

- For **Additive** models,

$$\text{LDFS} \quad = \quad \text{BOUNDED LDFS}$$

- For **Max** models,

$$\text{LDFS} \quad \neq \quad \text{BOUNDED LDFS}$$

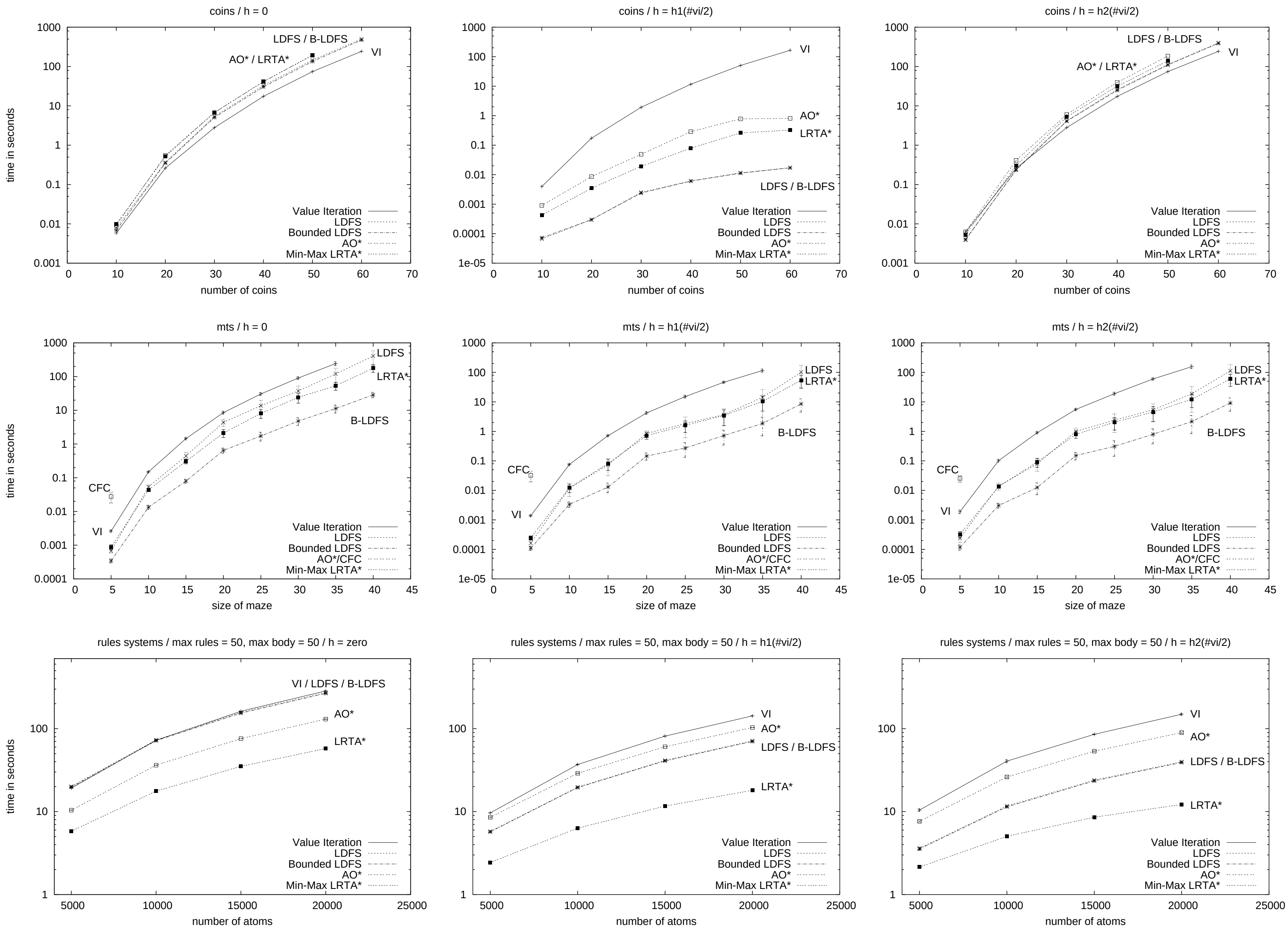
LDFS (like VI, AO*, min-max LRTA*, etc) computes optimal solutions graphs where each node is an optimal solution subgraph; over **Max Models**, this isn't needed. **Bounded LDFS fixed this, enforcing consistency only where needed**

Empirical Evaluation: Algorithms, Heuristics, Domains

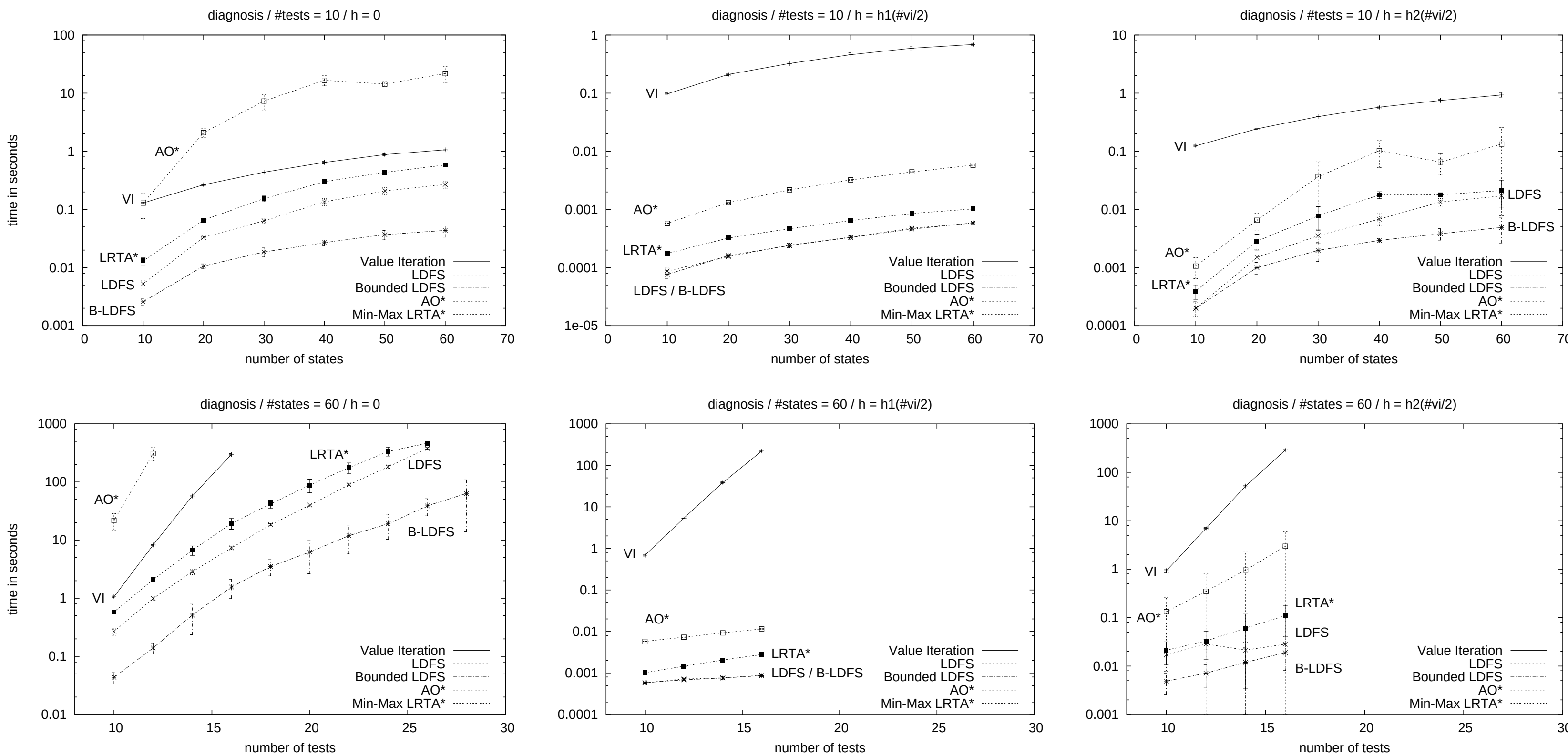
- **Algorithms:** VI, AO*/CFC$_{rev^*}$, min-max LRTA*, LDFS, BOUNDED LDFS
- **Heuristics:** $h = 0$ and two domain-independent heuristics h_1 and h_2
- **Domains**
  - **Coins:** Find counterfeit coin among N coins; $N = 10, 20, \ldots, 60$.
  - **Diagnosis:** Find true state of system among M states with N binary tests: In one case, $N = 10$ and M in $\{10, 20, \ldots, 60\}$, in second, $M = 60$ and N in $\{10, 12, \ldots, 28\}$.
  - **Rules:** Derivation of atoms in acyclic rule systems with N atoms, and at most R rules per atom and M atoms per rule body . . . $R = M = 50$ and N in $\{5000, 10000, \ldots, 20000\}$.
  - **MTS:** Predator must catch a prey that moves non-deterministically to a non-blocked adjacent cell in a given random maze of size $N \times N$; $N = 15, 20, \ldots, 40$ . . .

problem	$\|S\|$	V^*	N_{VI}	$\|A\|$	$\|F\|$	$\|\pi^*\|$
coins-10	43	3	2	172	3	9
coins-60	1,018	5	2	315K	3	12
mts-5	625	17	14	4	4	156
mts-35	1, 5M	573	322	4	4	220K
mts-40	2, 5M	684	–	4	4	304K
diag-60-10	29,738	6	8	10	2	119
diag-60-28	$>$ 15M	6	–	28	2	119
rules-5000	5,000	156	158	50	50	4,917
rules-20000	20,000	592	594	50	50	19,889

Empirical Evaluation: Results (1)



Empirical Evaluation: Results (2)



Runtimes are roughly BOUNDED LDFS$<$ LDFS$\leq$ LRTA*$<$ AO*$<$ VI, except in RULES where LRTA* is best.

Conclusions

- Unified computational framework, that is simple, general, and efficient

$$\textbf{LDFS} = \textbf{Depth First Search} + \textbf{Learning}$$

- Reduces to state-of-the-art algorithms in some models (OR Graphs and GTs)
- Yields new competitive algorithms in others (e.g., AND/OR Graphs)
- Shows that ideas underlying a wide range of algorithms **reduce to**:

Depth First Search
Lower Bounds
Learning