SetA*: A Family of Efficient Search Algorithms
Combining Symbolic and Heuristic Search

Rune M. Jensen
Department of Innovation
IT University of Copenhagen
2 March 2004

Joint work with Randal E. Bryant and Manuela M. Veloso
Computer Science Department
Carnegie Mellon University, Pittsburgh, USA
Motivation

Two efficient search approaches:
Motivation

Two efficient search approaches:

- Use heuristics to guide the search toward the goal states [e.g., A*, Hart et al. 68]

Represent and manipulate states implicitly with OBDDs [e.g., Symbolic Model Checking, McMillan 93]

No successful combination exists!
Motivation

Two efficient search approaches:

- Use heuristics to guide the search toward the goal states [e.g., $A^*$, Hart et al. 68]
- Represent and manipulate states implicitly with OBDDs [e.g., Symbolic Model Checking, McMillan 93]
Motivation

Two efficient search approaches:

- Use heuristics to guide the search toward the goal states [e.g., A*, Hart et al. 68]
- Represent and manipulate states implicitly with OBDDs [e.g., Symbolic Model Checking, McMillan 93]

No successful combination exists!
Motivation

Two efficient search approaches:

- Use heuristics to guide the search toward the goal states [e.g., A*, Hart et al. 68]
- Represent and manipulate states implicitly with OBDDs [e.g., Symbolic Model Checking, McMillan 93]

No successful combination exists!
Talk Outline

OBDDs  State-Set Branching  Heuristic Search  ghSetA*  fSetA*
Find a Route from MoMa to the Pond
2: Transitions

Central Park

The Pond
2: Transitions

The Pond
Central Park
2: Transitions

- Central Park
- The Pond
2: Transitions

The Pond

Central Park
2: Transitions

Central Park

The Pond
4: Goal States
Search Problem Solutions

Central Park

The Pond
Search Problem Solutions

Central Park

The Pond
Search Problem Solutions

The Pond

Central Park

Cost = 41
Search Problem Solutions

Central Park

The Pond

Cost = 9

Cost = 41
Optimal Solutions

The Pond

Central Park

Cost = 9
Search Algorithms
Search Algorithms

Central Park

The Pond

Expansion
0

RJ – p.12/41
Search Algorithms

The Pond

Central Park

Expansion

1
Search Algorithms
Search Algorithms

Central Park

The Pond

Expansion 3
Search Algorithms

The Pond

Central Park

Expansion 4
Search Algorithms

Central Park

The Pond

Expansion 5
Search Algorithms

Central Park

The Pond

Expansion
9

RJ – p.12/41
Dijkstra’s Algorithm
Dijkstra’s Algorithm

Expansion

The Pond

Central Park
Dijkstra’s Algorithm

Central Park

The Pond

Expansion

1
Dijkstra’s Algorithm
Dijkstra’s Algorithm

Central Park

The Pond

Expansion

3
Dijkstra’s Algorithm

![Diagram of Dijkstra's Algorithm with nodes and edges labeled. The Pond and Central Park are highlighted in green.]}
Dijkstra’s Algorithm

The Pond

Central Park

Expansion

4

0 1

2 3

4 5

6 7

8 9

10 11

> 11
Dijkstra’s Algorithm

The Pond

Central Park

Expansion

6

0 1
2 3
4 5
6 7
8 9
10 11
> 11
Dijkstra’s Algorithm

The Pond

Central Park

Expansion 10

0 1
2 3
4 5
6 7
8 9
10 11
> 11

RJ – p.13/41
Dijkstra’s Algorithm

Central Park

The Pond

Expansion

17

0 1

2 3

4 5

6 7

8 9

10 11

> 11
Dijkstra’s Algorithm

Central Park

The Pond

Expansion 23

0 1
2 3
4 5
6 7
8 9
10 11
> 11

The Pond
Dijkstra’s Algorithm

The Pond
Central Park

Expansion
32

0 1
2 3
4 5
6 7
8 9
10 11
> 11

a b c d e f g
Dijkstra’s Algorithm

Expansion 43

Central Park

The Pond

RJ – p.13/41
Dijkstra’s Algorithm

Central Park

The Pond
Symbolic Search
Binary Decision Diagrams (BDDs)

A BDD is a compact representation of a Boolean function
Binary Decision Diagrams (BDDs)

A BDD is a compact representation of a Boolean function

**Definition**
A BDD is a DAG with,

- One or two terminal nodes labeled 0 or 1,
- A set of variable nodes with two edges \( \text{low} \) and \( \text{high} \),
- A linear ordering of variables \( x_1, x_2, \ldots, x_n \).
1. Uniqueness of nodes associated to the same variable:
Reductions

1. Uniqueness of nodes associated to the same variable:

   \[ x \]
   \[ x \]

2. No redundant tests:

   \[ x \]
   \[ x \]

Result: A canonical representation!
A simple example

\[(x_1 \land y_1) \lor (x_2 \land y_2) \lor (x_3 \land y_3)\]
Exponential Blow Up with bad Variable Ordering

\[(x_1 \land y_1) \lor (x_2 \land y_2) \lor (x_3 \land y_3)\]
Building Expressions: base cases

- True, False \(O(1)\)
  - \textit{true} \hspace{1cm} 1
  - \textit{false} \hspace{1cm} 0

- Variables \(O(1)\)
  - \(x\)
    - 0
    - 1
Building Expressions: inductive cases

- **Binary operations** $O(|f||g|)$:

  $f \oplus g$
  
  $= (x_1 \rightarrow f_{\text{high}}, f_{\text{low}}) \oplus (x_1 \rightarrow g_{\text{high}}, g_{\text{low}})$
  
  $= x_1 \rightarrow (f_{\text{high}} \oplus g_{\text{high}}), (f_{\text{low}} \oplus g_{\text{low}})$
  
  $= \ldots$

- **Unary operations** $O(|f|)$:

  Implemented with binary operations
BDD-based Search

Key ideas [McMillan 93]:

- Encode states as Boolean vectors
- Use BDDs to
  - Represent sets of states
  - Represent the transition relation
  - Perform the search
BDD Representation of an Example

\[ G \]

\[ \begin{array}{c}
(0,1) \quad \rightarrow \quad (1,1) \\
\uparrow \quad \downarrow \\
(0,0) \quad \leftrightarrow \quad (1,0)
\end{array} \]

\[ S_0 \]

\[ s_0 \quad = \quad \neg v_0 \land \neg v_1 \]

\[ T(v_0, v_1, v'_0, v'_1) \quad = \quad \neg v_0 \land \neg v_1 \land v'_0 \land \neg v'_1 \]

\[ \lor \quad \neg v_0 \land \neg v_1 \land \neg v'_0 \land v'_1 \]

\[ \lor \quad \neg v_0 \land v_1 \land v'_0 \land v'_1 \]

\[ \lor \quad v_0 \land \neg v_1 \land \neg v'_0 \land \neg v'_1 \]

\[ \lor \quad v_0 \land v_1 \land v'_0 \land \neg v'_1 \]

RJ – p.22/41
Computing the Search Frontier

- Frontier computation = Image computation

\[
\text{image}(S(\bar{v})) = \left( \exists \bar{v} \cdot S(\bar{v}) \land T(\bar{v}, \bar{v}') \right)[\bar{v}' / \bar{v}]
\]

- Example \( S(v_0, v_1) = \neg v_0 \land \neg v_1 \)

\[
\text{image}(S(v_0, v_1)) \\
= (\exists (v_0, v_1) \cdot \neg v_0 \land \neg v_1 \land T(v_0, v_1, v'_0, v'_1))[(v'_0, v'_1)/(v_0, v_1)] \\
= (v'_0 \land \neg v'_1 \lor \neg v'_0 \land v'_1)[(v'_0, v'_1)/(v_0, v_1)] \\
= v_0 \land \neg v_1 \lor \neg v_0 \land v_1
\]
Disjunctive Partitioning

\[ T(\bar{v}, \bar{v}') = P_1(\bar{v}, \bar{v}') \lor P_2(\bar{v}, \bar{v}') \lor \cdots \lor P_n(\bar{v}, \bar{v}') \]
Disjunctive Partitioning

- Disjunctive partitioning

\[ T(\bar{v}, \bar{v}') = P_1(\bar{v}, \bar{v}') \lor P_2(\bar{v}, \bar{v}') \lor \cdots \lor P_n(\bar{v}, \bar{v}') \]

- Image computation for a disjunctive partitioning

\[ \text{image}(S(\bar{v})) = \bigvee_{i=1}^{\vert P \vert} \left( \exists \bar{v} \, . \, S(\bar{v}) \land P_i(\bar{v}, \bar{v}') \right)[\bar{v}' / \bar{v}] \]
Heuristic Search

OBDDs
State-Set Branching
Heuristic Search
Main idea in Heuristic Search

Use a **heuristic estimate** of the distance to the states you want to reach. Use the estimate to **guide** the search in that direction.

*Current paths explored*

*States we want to reach*
Introduction to Heuristic Search (A*)

The Pond

Central Park

RJ – p.27/41
Introduction to Heuristic Search (A*)

The Pond

Central Park

RJ – p.27/41
Introduction to Heuristic Search (A*)

Cost of reaching node: 0

Central Park

The Pond

Expansion

0

0 1 2 3 4 5 6 7 8 9 10 11

> 11
Introduction to Heuristic Search (A*)

Cost of reaching node: 0
Manhattan Distance to goal: 5
Introduction to Heuristic Search (A*)

Cost of reaching node: 0
Manhattan Distance to goal: 5
f = g + h

The Pond
Central Park

RJ – p.27/41
Introduction to Heuristic Search (A*)

Expansion

Cost of reaching node: 2

Central Park
The Pond

Cost of reaching node: > 11

RJ – p.27/41
Introduction to Heuristic Search (A*)

Cost of reaching node: 2
Manhattan Distance to goal: 7

Expansion 1
Cost of reaching node: \( g \)
Manhattan Distance to goal: \( h \)
\[ f = g + h \]
Introduction to Heuristic Search (A*)

The Pond
Central Park

Expansion
1

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|----|----|----|
| a |   |   |   |   |   |   |   |   |   |   |    |    |    |
| b |   |   |   |   |   |   |   |   |   |   |    |    |    |
| c |   |   |   |   |   |   |   |   |   |   |    |    |    |
| d |   |   |   |   |   |   |   |   |   |   |    |    |    |
| e |   |   |   |   |   |   |   |   |   |   |    |    |    |
| f |   |   |   |   |   |   |   |   |   |   |    |    |    |
| g |   |   |   |   |   |   |   |   |   |   |    |    |    |

Colors:
- 0: White
- 1: Gray
- 2: Yellow
- 3: Orange
- 4: Red
- 5: Purple
- 6: Blue
- 7: Green
- >11: Black

RJ – p.27/41
Introduction to Heuristic Search (A*)

The Pond

Central Park

Expansion

2

0 1
2 3
4 5
6 7
8 9
10 11

> 11

RJ – p.27/41
Introduction to Heuristic Search (A*)

The Pond
Central Park

Expansion
3

RJ – p.27/41
Introduction to Heuristic Search (A*)

Expansion
10

Central Park

The Pond
Introduction to Heuristic Search (A*)
Introduction to Heuristic Search (A*)
Combining Symbolic and heuristic Search

ghSetA*
fSetA*
Combining Symbolic and heuristic Search

Previous Attempts

- **In formal verification**
  - [Yuan et al., CAV’97]
  - SpotLight [Yang & Dill, DAC’98]

  Limitations: 2 phase expansion, specialized algorithms

- **In AI**
  - BDDA* [Edelkamp & Reffel 98]
  - (ADDA* [HZF02])

  complex arithmetic operations, mixed semantics at BDD level
State-Set Branching: part I

Associate change in search node information with transitions

\[ h(b) = h(a) + \Delta h \]

\[ h(b) = h(b) - h(a) \]

\[ \Delta h = h(b) - h(a) \]

\[ h(a) \]

\[ h(b) \]

a

b
State-Set Branching: part I

Associate change in search node information with transitions

\[ \Delta h = h(b) - h(a) \quad h(b) = h(a) + \Delta h \]

\[ \Delta f = c + h(b) - h(a) \quad f_b = f_a + \Delta f \]
State-Set Branching: part I

Associate change in search node information with transitions

\[ \Delta h = h(b) - h(a) \]
\[ h(b) = h(a) + \Delta h \]

\[ \Delta f = c + h(b) - h(a) \]
\[ f_b = f_a + \Delta f \]

\[ \Delta f = (c, h(b) - h(a)) \]
\[ (g_b, h_b) = (g_a, h_a) + \Delta f \]
State-Set Branching: part II

Partition the transitions according to change in search node information

\[ T(s, s') = T_{\Delta I_1}(s, s') \lor \cdots \lor T_{\Delta I_m}(s, s') \]

Disjunctive Branching partitioning
State-Set Branching: part II

Often possible to compute branching partitioning directly from the action description

No explicit enumeration of transitions!
State-Set Branching: part II

Otherwise use a symbolic computation of the search information change

$$\Delta h(\bar{v}, \bar{v}', \bar{\Delta}) = h(\bar{e}, \bar{v}) \land h(\bar{e}', \bar{v}') \land (\bar{\Delta} = \bar{e}' - \bar{e})$$

Again, no explicit enumeration of transitions!
State-Set Branching: part III

Find next states and propagate search information in one phase

\[ T_{\Delta I_1} \quad T_{\Delta I_2} \quad \ldots \quad T_{\Delta I_m} \]

\[ I + \Delta I_1 \quad I + \Delta I_2 \quad \ldots \quad I + \Delta I_m \]
SetA* [Jensen, Veloso, Bryant, 02]
SetA* [Jensen, Veloso, Bryant, 02]

Expansion 1

The Pond
Central Park

RJ – p.35/41
The Pond

Central Park

Expansion

2

Legend:
- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- > 11
SetA* [Jensen, Veloso, Bryant, 02]

```
<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Expansion 8
```

The diagram shows a grid with nodes labeled from 0 to 15, representing a search space. The grid is divided into sections labeled Central Park and The Pond. The expansion of nodes is shown with arrows indicating the connections and costs. The colors of the nodes represent different cost values, with 0 being the lowest cost and 15 being the highest.
The SetA* Algorithms [JBV, AAAI’02]

▷ **FSetA***
- Evaluation function $f(n) = g(n) + h(n)$
- $I_0 = h(s_0)$
- $\Delta I(s, s') = c(s, s') + h(s') - h(s)$

▷ **GSetA***
- Evaluation function $f(n) = g(n) + h(n)$
- $I_0 = (0, h(s_0))$
- $\Delta I(s, s') = (c(s, s'), h(s') - h(s))$
Experimental Results: $FG^k$
Experimental Results: $FG^k$
Experimental Results: $F G^k$
$F G^k (n=16)$

FGk, n=16, $|\text{space}|=2^{33}$

Total CPU time (log scale)

Number of unguided steps (k)
Search and Planning Results

HSPr
Backward
Blocks World

HSPr
Backward
Logistics

Manhattan distance
Forward
24-Puzzle

Hamming distance
Forward
DVM Puzzle
Search and Planning Results

Blocks AIPS-00 (HSPr)

Logistics AIPS-00 (HSPr)

24-Puzzle

DV4M15 |space| = 2^{60}
State-Set Branching

- A **general framework** for combining BDD-based and heuristic search
  - Any BFS algorithm
  - Any heuristic function
  - Any evaluation function
  - Transition cost function

- **High performance**
  - \textsc{seta} often dominates both blind BDD-based search and A*
  - Consistently outperforms BDDA* and ADDA*