University of Basel

Implementation and Evaluation of Depth-First IBEX in Fast Downward

Petr Sabovčik <petr.sabovcik@stud.unibas.ch>

Department of Mathematics and Computer Science, University of Basel

30.07.2024

Background

State Spaces

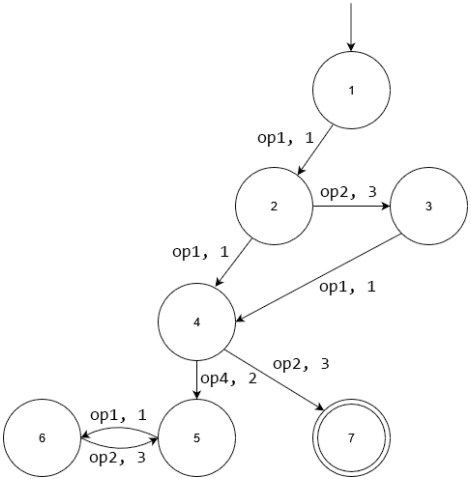
Includes:

- An initial state:

5	4	3
2	1	

- A goal state:

	1	2
3	4	5



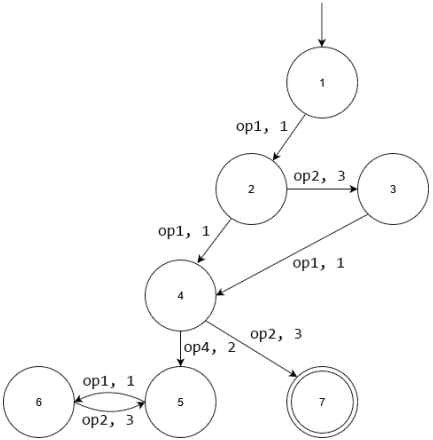
Depth-First Search (DFS)

- Uninformed
- Explores as far as possible along each branch before backtracking
- Not optimal

Depth-First Search - Outline

1. Check if the current state is the goal state and return the path if it is
2. If not, move to a successor state and repeat, backtrack if no more successors

Depth-First Search



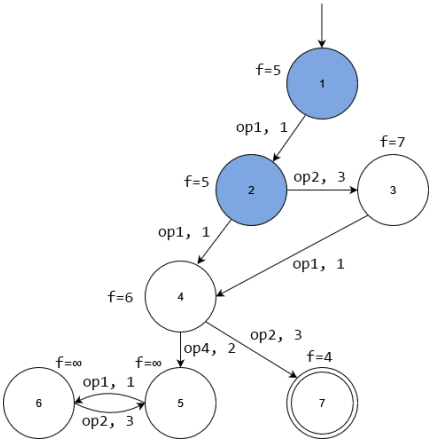
Iterative Deepening A* (IDA*)

- Informed
- Iterative-deepening approach
- Combination of DFS and A*
- Optimal

Iterative Deepening A* (IDA*)

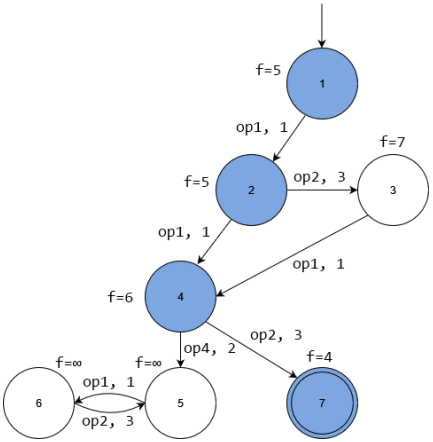
1. Set f-bound to $h(initial\_state)$
2. Start f-bouned DFS
   - 2.1 Check if the current state is the goal state and return the path if it is
   - 2.2 If the f-bound is exceeded, backtrack
   - 2.3 Move to a successor state and repeat, backtrack if no more successors
3. Repeat with a higher f-bound if a solution was not found

IDA*



f-bound $= h(initial\_state) = 5$

IDA*



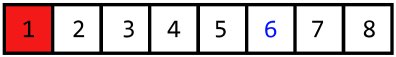
f-bound = 6, solution found: op1, op1, op2

Depth-First IBEX

Depth-First IBEX Overview

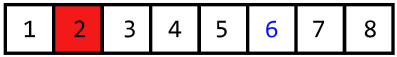
- Depth-First IBEX or Budgeted Tree Search (BTS) is an informed iterative-deepening search algorithm
- Enforces each iteration to consider exponentially more nodes
- Optimal
- Uses exponential search

Exponential Search



Exponential Search

Exponential Search

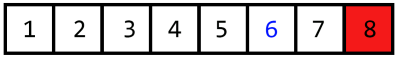


Exponential Search Phase, $i = 0 + 2^1 = 2$

Exponential Search

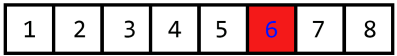
Exponential Search Phase, $i = 0 + 2^2 = 4$

Exponential Search



Exponential Search Phase, $i = 0 + 2^3 = 8$

Exponential Search



Binary Search Phase, $i = (4 + 8)/2 = 6$

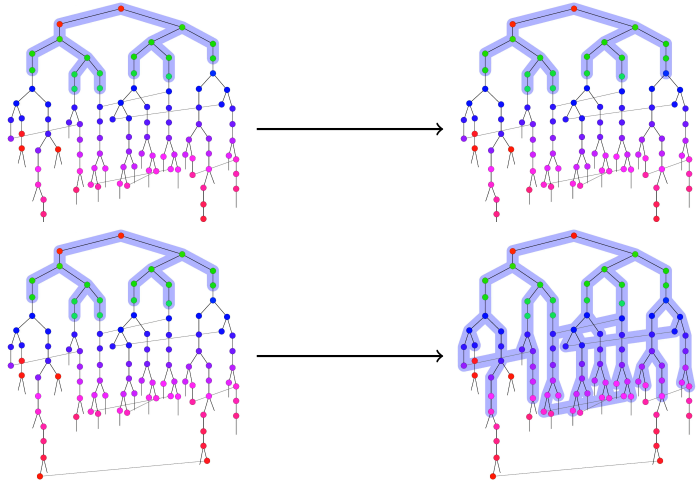
Depth-First IBEX

1. Set f-bound to $h(initial\_state)$
2. Start f-bouned DFS
   - 2.1 Check if the current state is the goal state and return the path if it is
   - 2.2 If the f-bound is exceeded, backtrack
   - 2.3 Move to a successor state and repeat
3. Repeat with a higher f-bound if a solution was not found and the number of expansions is higher than desired growth rate
4. Enter exponential search phase
   - 4.1 Grow f-bound exponentially until budget hit or exhausted
   - 4.2 If budget is exhausted, find an f-bound within the budget using binary search
5. Repeat until a proven optimal solution is found

Depth-First IBEX Properties

- Worst-case time complexity: $O(Nlog(C^*/\epsilon))$
  - C^* - optimal solution cost
  - ϵ - granularity of action costs
  - With a linear growth of expansions in each iteration IDA* has a worst-case time complexity of $\Theta(N^2)$
- Space complexity: $O(bd)$
  - b - branching factor
  - d - depth of the optimal solution
  - Same as IDA*

Depth-First IBEX Example - Difference between iterations



Depth-First IBEX Example - Animation

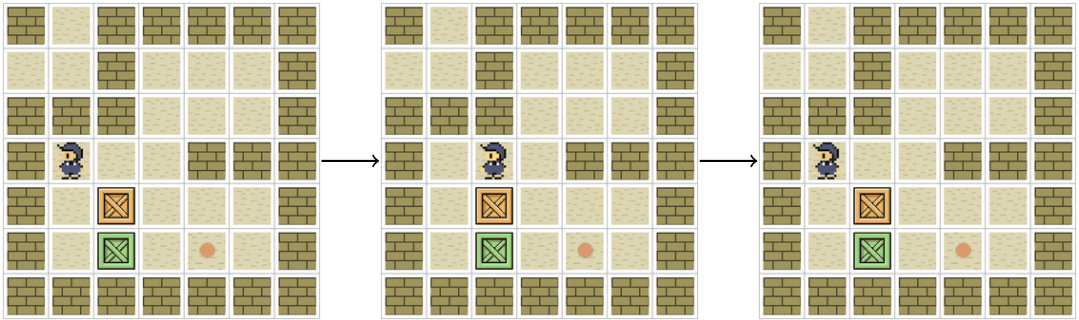
https://www.movingai.com/SAS/BTS/BTS.mp4

Implementation

Implementation

- Implemented as a `SearchAlgorithm` in Fast Downward
- Requires `cache_estimates` option to be disabled
- Path checking option - disables duplicate states in one path

Path Checking



Evaluation

Evaluation - $h^{\text{LM-CUT}}$

Algorithm Name	A*	BTS	BTS path checking	IDA*	IDA* path checking
Coverage	**966**	559	596	556	591
Exponential search	—	13.82%	17.88%	—	—
Expansions until last jump	—	978.25	1629.09	**767.11**	1091.47
# Iterations	—	**1757**	1988	1821	2188
Search time (s)	**0.09**	0.77	0.54	0.70	0.49

Evaluation - h^{blind}

Algorithm Name	A*	BTS	BTS path checking	IDA*	IDA* path checking
Coverage	**718**	257	287	246	270
Exponential search	—	19.21%	23.17%	—	—
Expansions until last jump	—	171745.17	121924.19	127070.28	**54197.04**
# Iterations	—	**2158**	2398	3075	3316
Search time (s)	**0.02**	1.11	0.53	1.43	0.62

Differences in Coverage - $h^{\text{LM-CUT}}$

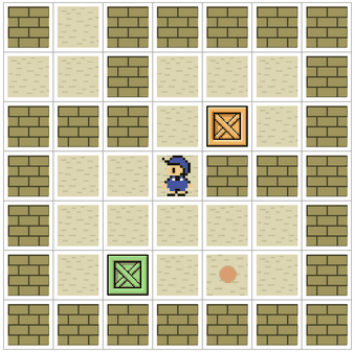
Domain	BTS	IDA*
mprime$^{(35)}$	**21**	20
nomystery-opt11-strips$^{(20)}$	11	**12**
organic-synthesis-split-opt18-strips$^{(20)}$	**14**	13
parcprinter-08-strips$^{(30)}$	**14**	13
parcprinter-opt11-strips$^{(20)}$	**9**	8

Differences in Coverage - h^{blind}

Domain	BTS	IDA*
movie$^{(30)}$	**3**	2
organic-synthesis-split-opt18-strips$^{(20)}$	**10**	9
parcprinter-08-strips$^{(30)}$	**5**	3
parcprinter-opt11-strips$^{(20)}$	**2**	0
pegsol-08-strips$^{(30)}$	**26**	24
pegsol-opt11-strips$^{(20)}$	**16**	14
pipesworld-notankage$^{(50)}$	**6**	5

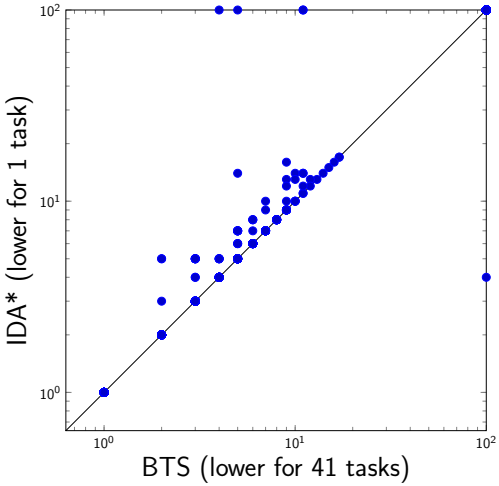
Stack Overflows

- Stack overflows occurred as result of too-deep recursion
- Happens in certain problems with 0 cost actions
- f-bound never met
- Avoided with path checking

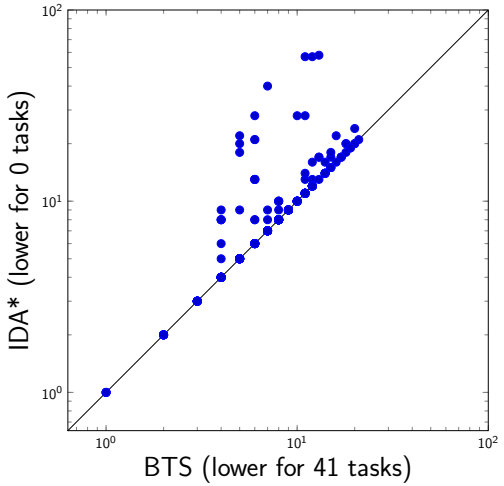


Sokoban

Number of Iterations - $h^{\text{LM-CUT}}$



Number of Iterations - h^{blind}



Conclusion

Conclusion

- BTS improves on the number of iterations required
- Requires more expansions
- A* better suited for the IPC benchmark suite

Questions?

petr.sabovcik@stud.unibas.ch