

Foundations of Artificial Intelligence March 27, 2023 — 17. State-Space Search: IDA*		
17.1 IDA*: Idea		
17.2 IDA*: Algorithm		
17.3 IDA*: Properties		
17.4 Summary		
T. Keller & F. Pommerening (University of B Foundations of Artificial Intelligence	March 27, 2023	2 / 19



T. Keller & F. Pommerening (University of B Foundations of Artificial Intelligence

17. State-Space Search: IDA*

IDA*: Idea

March 27, 2023 5 / 19

17

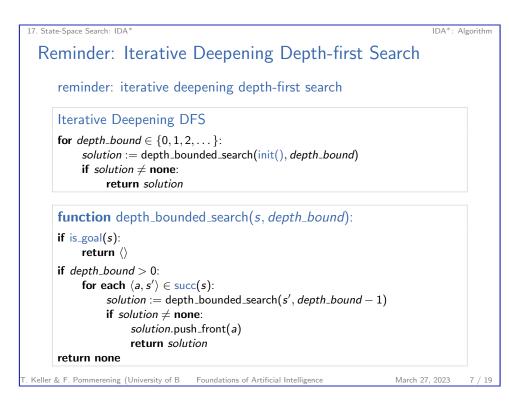
IDA*

The main drawback of the presented best-first graph search algorithms is their space complexity.

Idea: use the concepts of iterative-deepening DFS

- bounded depth-first search with increasing bounds
- ▶ instead of depth we bound *f* (in this chapter f(n) := g(n) + h(n.state) as in A^{*})
- \rightarrow IDA* (iterative-deepening A*)
- tree search, unlike the previous best-first search algorithms

Keller & F. Pommerening (University of B Foundations of Artificial Intelligence



17. State-Space Search: IDA*		IDA*: Algorith
17.2 IDA*: AI	gorithm	
	0	
Keller & F. Pommerening (University of B Fo	undations of Artificial Intelligence	March 27, 2023 6 / 1

First Attempt: IDA* Main Function first attempt: iterative deepening A* (IDA*) IDA* (First Attempt) for $f_bound \in \{0, 1, 2,\}$: $solution := f_bounded_search(init(), 0, f_bound)$ if $solution \neq$ none: return $solution$	State-Space Search: IDA*	IDA*: Algorithm
IDA* (First Attempt) for $f_bound \in \{0, 1, 2,\}$: solution := $f_bounded_search(init(), 0, f_bound)$ if solution \neq none:	First Attempt: IDA* Main Function	
IDA* (First Attempt) for $f_bound \in \{0, 1, 2,\}$: solution := $f_bounded_search(init(), 0, f_bound)$ if solution \neq none:		
$IDA^* (First Attempt)$ for f_bound $\in \{0, 1, 2,\}$: solution := f_bounded_search(init(), 0, f_bound) if solution \neq none:		
for $f_bound \in \{0, 1, 2,\}$: $solution := f_bounded_search(init(), 0, f_bound)$ if $solution \neq$ none:	first attempt: iterative deepening A^* (IDA*)	
$solution := f_bounded_search(init(), 0, f_bound)$ if $solution \neq none$:	IDA* (First Attempt)	
if solution \neq none:		
return solution		
	return solution	

17. State-Space Search: IDA* IDA*: Algorithm First Attempt: *f*-Bounded Search **function** f_bounded_search(*s*, *g*, *f_bound*): if $g + h(s) > f_{-bound}$: return none if is_goal(s): return () for each $\langle a, s' \rangle \in \text{succ}(s)$: $solution := f_bounded_search(s', g + cost(a), f_bound)$ if solution \neq none: *solution*.push_front(*a*) return solution return none Keller & F. Pommerening (University of B Foundations of Artificial Intelligence March 27, 2023 9 / 19

17. State-Space Search: IDA*

Growing the *f* Bound

In IDDFS, we grow the bound from the smallest bound that gives a non-empty search tree (0) by 1 at a time.

- This usually leads to exponential growth of the tree between rounds, so that re-exploration work can be amortized.
- In our first attempt at IDA*, there is no guarantee that increasing the f bound by 1 will lead to a larger search tree than in the previous round.
- This problem becomes worse if we also allow non-integer (fractional) costs, where increasing the bound by 1 would be very arbitrary.



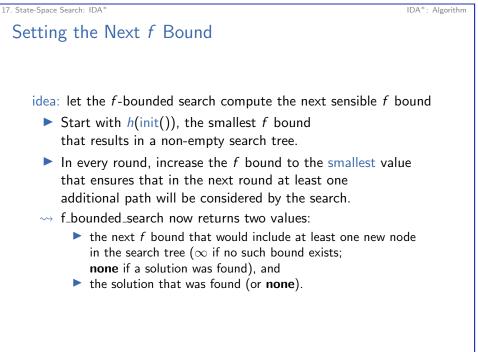
IDA* First Attempt: Discussion

- The pseudo-code can be rewritten to be even more similar to our IDDFS pseudo-code. However, this would make our next modification more complicated.
- The algorithm follows the same principles as IDDFS, but takes path costs and heuristic information into account.
- For unit-cost state spaces and the trivial heuristic h : s → 0 for all states s, it behaves identically to IDDFS.
- For general state spaces, there is a problem with this first attempt, however.

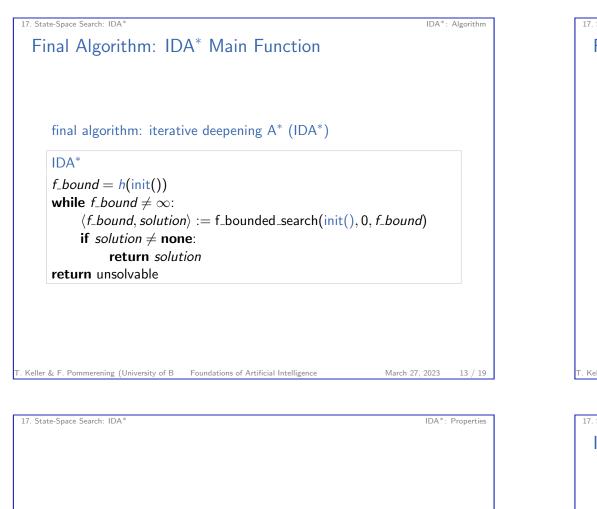
Keller & F. Pommerening (University of B Foundations of Artificial Intelligence

March 27, 2023 10 / 19

IDA*: Algorithm



IDA*: Algorithm

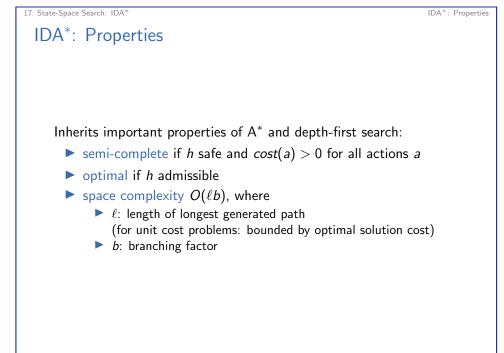


17.3 IDA*: Properties

17. State-Space Search: IDA*

Final Algorithm: *f*-Bounded Search

$g + h(s) > f_{-}bound$	
return $\langle g + h(s),$	none〉
is_goal(s):	
return $\langle none, \langle \rangle \rangle$	
ew_bound := ∞	
or each $\langle a,s' angle\insucc$	
A CONTRACT OF	$ tion angle := f_bounded_search(s', g + cost(a), f_bound)$
if solution \neq none	
<i>solution</i> .push	
return (none	
	n(new_bound, child_bound)
eturn (<i>new_bound</i> , no	ne>



IDA*: Algorithm

17. State-Space Search: IDA*

IDA*: Properties

17. State-Space Search: IDA*

IDA*: Discussion

 compared to A* potentially considerable overhead because no duplicates are detected

- \rightsquigarrow exponentially slower in many state spaces
- → often combined with partial duplicate elimination (cycle detection, transposition tables)
- overhead due to iterative increases of f bound often negligible, but not always
 - especially problematic if action costs vary a lot: then it can easily happen that each new f bound only considers a small number of new paths

F. Keller & F. Pommerening (University of B Foundations of Artificial Intelligence

17. State-Space Search: IDA*
Summary
IDA* is a tree search variant of A* based on iterative deepening depth-first search
main advantage: low space complexity
disadvantage: repeated work can be significant
most useful when there are few duplicates

17.4 Summary

. Keller & F. Pommerening (University of B Foundations of Artificial Intelligence

Summary

March 27, 2023

18 / 19

. Keller & F. Pommerening (University of B Foundations of Artificial Intelligence

March 27, 2023 19 / 19

March 27, 2023

17 / 19

Summary