



EEL5840: Elements of Machine Intelligence

Robobug

Announcements



- Reading Assignment:
 - > Nilsson chapter 9
- Announcements:
 - > 1st Exam Date
Thu. Oct. 8, 2009 in class
- Today's Handouts in WWW:
 - > Outline Class 19
- Web Site
 - > www.mil.ufl.edu/eel5840
 - > Software and Notes
 - > XLISP Documentation



University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

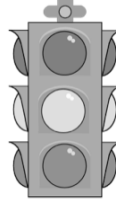
1

EEL5840: Elements of Machine Intelligence

Robobug

Today's Menu

- Uninformed Search (Chapter 8)
 - ⇒ Formulating the State Space
 - ⇒ Components of Implicit State Graphs
 - ⇒ Backtracking
 - ⇒ Backtracking & Depth-First Search



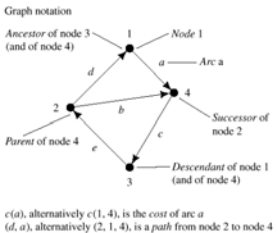
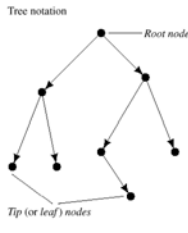
University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

2

EEL5840: Elements of Machine Intelligence

Robobug

Summary of Graph and Tree Notation

Graph notation

Node 1

Node 2

Node 3

Node 4

Arc: a

Successor of node 2

Parent of node 4

Descendant of node 1 (and of node 4)

Tip (or leaf) nodes

c(a), alternatively $c(1, 4)$, is the cost of arc a
 (d, a), alternatively (2, 1, 4), is a path from node 2 to node 4

Figure 7.5 Graph and Tree Notation

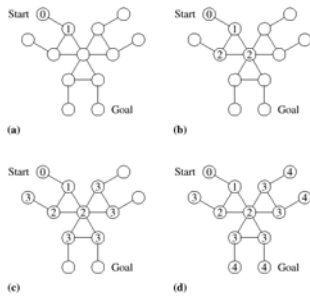
University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

3

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies



Start (0)

Goal

(a)

(b)

(c)

(d)

Figure 7.3 Stages of Search

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

4

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- The Representation Problem: dealing w/ large state space
 - Careful formulation is required - choose your representation, iconic model or data structure by taking inventory of the problem domain and any other problem-specific knowledge
 - Methods are required to represent large graphs implicitly
 - Efficient search methods are required

2	8	3
1	6	4
7		5

→

1	2	3
8		4
7	6	5

Eight Puzzle State Space
 $O(9!) = 362,880$ nodes

In this problem the obvious(?) iconic representation is a 3x3 array, with the blank tile moving {up, down, left, right}

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Aravamudan

EEL5840: Elements of Machine Intelligence

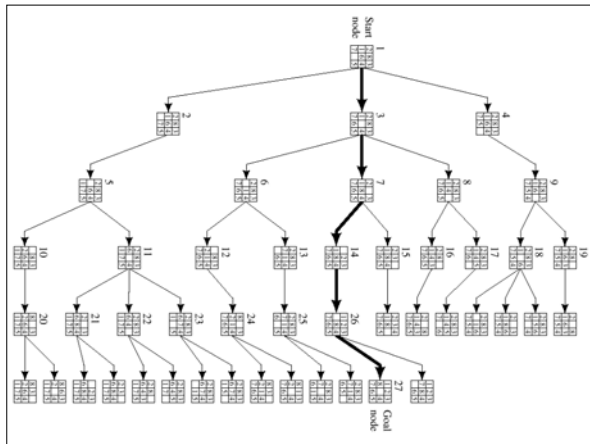
Robobug

Search Strategies

- The Representation Problem
 - There are 24 rules possible to legally move a tile: {4 moves from the middle, 2 moves for each of the 4 corners, 3 moves from each of the 4 centers of rows/cols}
 - Alternatively we have 4 rules to move the blank: {left, right, up or down} {L,R,U,D}

Moves from center or from any middle row/col.

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Aravamudan



EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- Backtracking : A recursive procedure that is adequate for problems requiring small amounts of search, has a simple programming model, and requires small memory space.

Notes on BACKTRACK{,1}

- The BACKTRACK procedures are recursive
- The number of arguments is 1
 - BACKTRACK the type is the same as the global DB
 - BACKTRACK1 the type is a list of DBs on a path back to the initial DB
- Returns a list of rules that if applied in sequence satisfies the termination condition or fail
- BOUND implements a depth bound check

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Aravamudan

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- Recursive Procedure BACKTRACK(DATA)
 1. if term(DATA) return nil
 2. if deadend(DATA) return fail
 3. RULES ← Apprules(DATA)
 4. LOOP: if null(RULES) return fail
 5. R ← first(RULES)
 6. RULES ← rest(RULES)
 7. RDATA ← R(DATA)
 8. PATH ← BACKTRACK(RDATA)
 9. if PATH=fail go LOOP
 10. return cons(R,PATH)

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

9

EEL5840: Elements of Machine Intelligence

Robobug

Backtracking Example

①	②	③	
2 8 3	2 8 3	2 8 3	
1 6 4	⇒⇒⇒ 1 6 4 ⇒⇒⇒	_ 6 4	{L,U,R,D}
7 _ 5	_ 7 5	1 7 5	

④	⑤	⑥	
_ 8 3	8 _ 3	_ 8 3	
2 6 4	⇒⇒⇒ 2 6 4 ⇒⇒⇒	2 6 4	
1 7 5	1 7 5	1 7 5	

Here the configuration is identical to No. 4, ∴ backtrack

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

10

EEL5840: Elements of Machine Intelligence

Robobug

Backtracking Example

⑤	⑥	⑦	
8 _ 3	8 3 _	8 _ 3	
2 6 4	⇒⇒⇒ 2 6 4 ⇒⇒⇒	2 6 4	Here the configuration is identical to No. 5, ∴ backtrack
1 7 5	1 7 5	1 7 5	

⑥	⑦	⑤	
8 3 _	8 3 4	8 _ 3	
2 6 4	⇒⇒⇒ 2 6 _ ⇒⇒⇒	2 6 4	{L,U,R,D}
1 7 5	1 7 5	1 7 5	

Here we have applied ≥ 6 rules ∴ backtrack

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

11

EEL5840: Elements of Machine Intelligence

Robobug

Backtracking Example

①	②	③	
2 8 3	2 8 3	2 _ 3	
1 6 4	⇒⇒⇒ 1 _ 4 ⇒⇒⇒	1 8 4	{U,D,L,R}
7 _ 5	7 6 5	7 6 5	

④	③	④	
2 8 3	2 _ 3	_ 2 3	
1 _ 4	⇒⇒⇒ 1 8 4 ⇒⇒⇒	1 8 4	
7 6 5	7 6 5	7 6 5	

Here the configuration is identical to No. 2, ∴ backtrack to No 3

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato Amato

12

EEL5840: Elements of Machine Intelligence

Robobug

Backtracking Example

① ② ③
 2 8 3 2 8 3 2 8 3
 1 6 4 ==> 1 6 4 ==> 1 6 4
 7 _ 5 _ 7 5 7 _ 5

{L,R,U,D}

Here the configuration is identical to No. 1, ∴ backtrack to No 2

② ③ ④
 2 8 3 2 8 3 2 8 3
 1 6 4 ==> _ 6 4 ==> 6 _ 4
 _ 7 5 1 7 5 1 7 5

Clearly this is going nowhere

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato, Amato

13

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- Recursive Procedure BACKTRACK1(DATALIST)
 - DATA ← first(DATALIST)
 - if member(DATA, rest(DATALIST)) return fail
 - if term(DATA) return nil
 - if deadend(DATA) return fail
 - if length(DATALIST) > BOUND return fail
 - RULES ← Apprules(DATA)
 - LOOP: if null(RULES) return fail
 - R ← first(RULES)
 - RULES ← rest(RULES)
 - RDATA ← R(DATA)
 - RDATALIST ← cons(RDATA,DATALIST)
 - PATH ← BACKTRACK1(RDATALIST)
 - if PATH=fail go LOOP
 - return cons(R,PATH)

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato, Amato

14

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- Comments on BACKTRACK{,1}
 - > Apprules uses heuristics to order the rules (either arbitrarily or by using domain knowledge). If you do not order the rules you will get lots of backtracking and be very inefficient.
 - > BOUND is a global variable based on an estimate of the number of levels required to reach a solution.
 - > BACKTRACK1 avoids cycles by checking all candidate DBs to see that they do not occur in a path back to the initial state.
 - > Notice how “close” to LISP is the specification of BACKTRACK{,1}

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato, Amato

15

EEL5840: Elements of Machine Intelligence

Robobug

Search Strategies

- Comments/Questions on BACKTRACKing
 - > You MUST use a depth bound (global variable BOUND) or you may backtrack an infinite number of times without arriving at a solution.
 - > How do you “skip” levels in order to arrive at a solution?
 - > Is LISP capable of handling complicated data structures?
 - > How much CPU time and memory is consumed by recursive implementations?
 - > How do we arrive at a reasonable depth bound?
- For example in the 8-puzzle
 - 2 8 3 Notice that we have W=4 tiles out of place.
 - 1 6 4 However, the actual number of moves required
 - 7 _ 5 is five and not four. Use BOUND={1.5,2}×W

University of Florida
EEL 5840 - Class #19 - Fall 2009
© Dr. A. Amato, Amato

16

