St. Francis Xavier University

CSCI-564
CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 13 – STATE SPACE PRUNING (CONTINUED)

Dr. Jean-Alexis Delamer

Recap

- Pruning is a technique to ignore parts of the search tree (and thus reduce the branching factor) to save runtime and memory.
- Pruning requires runtime and memory. We need to ensure that the costs are outweighed by the corresponding savings.
- Pruning exploit the expert knowledge of the domain.
  - Regularities.
- The pruning can be static or dynamic.

State Space Pruning

- Static and dynamic pruning give optimal solutions.
- The pruning algorithm needs to verify before pruning that the branch is not leading to an optimal solution.
- In large state space, pruning techniques does not reduce the time complexity enough.

What can we do?

Nonadmissible State Space Pruning

- We can sacrifice the optimality of the solution.
  - Sometimes a good, but quick solution is better.

- Example (GPS).
  - On small distances, you can calculate the optimal solution very fast.
  - But calculating the optimal path between Antigonish and San Francisco can be very long.
  - However, you only want a good solution not the optimal.
    - What is 1 hour on a 3-day travel.

Nonadmissible State Space Pruning

- The pruning technique that sacrifice the optimality are nonadmissible.
- We will see two techniques:
  - Macro problem solving
  - Relevance cut

Macro Problem Solving

- The idea is to group a sequence of actions into a new action.
  - Ex: 4x turn 90° can be grouped into turn 360°.
- The problem solver (algorithm) can apply multiple primitive operators at once.
- Where is the pruning?
  - Requires fewer decisions.
  - Choices inside a branch are ignored.

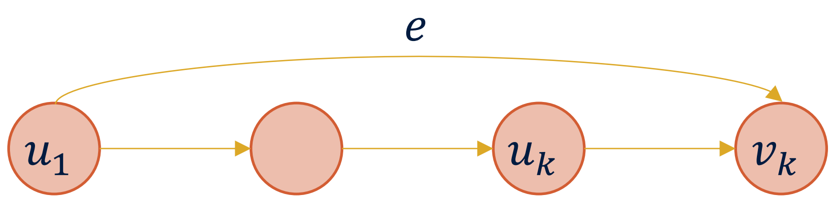
Macro Problem Solving

- Is there a catch?
  - If the substitutions operators are too generous (grouping to many primitive operators) the goal might not be found.

- We need to ensure that the goal is still reachable.

Macro Problem Solving

- Definition (Macro Operator):
  - A macro operator (macro) is a fixed sequence of elementary operators executed together.
  - For a problem graph with node set V and an edge set E.
  - A macro refers to an additional edge $e = (u, v)$ in $V \times V$ for which there are edges $e_1 = (u_1, v_1), \dots, e_k = (u_k, v_k) \in E$ with $u = u_1, v = v_k$ and $v_i = u_{i+1}$ for all $1 \leq i \leq k-1$.
- In other words, the path $(u_1, \dots, u_k, v_k)$ between u and v is shortcut by introducing e.



Macro Problem Solving

- Macros turn an unweighted graph into a weighted graph.
- Why?
  - Macros can have different lengths.
  - We need to know the weight of a macro to find the best solution.
- The weight of the macro is the accumulated weight of the original edges:
  - $w(u, v) = \sum_{i=1}^{k} w(u_i, v_i)$

Macro Problem Solving

- Inserting edges does not affect the reachability status of nodes.
- If there is no alternative in the choice of successors.
  - $Succ(u_i) = \{v_i\}$
  - Macros can substitute the original edges without loss of information.

- Example:
  - Maze areas with width of one (tunnel).

Macro Problem Solving

- If there are more paths between a node?
- To preserve the optimality of an underlying search algorithm.
  - We take the shortest path $w(u, v) = \delta(u, v)$.

- These macros are called admissible.

Macro Problem Solving

- How can we create the macros?
  - The *All-Pairs Shortest Paths* algorithm of Floyd-Warshall is one way.
  - At the end of the algorithm, all two nodes are connected.
  - The original edges are no longer needed to determine the shortest path.
  - It keeps the optimality of the search.

- So we can find the optimal solution with macros?

Macro Problem Solving

- True, for small problems.
- For larger problems computing *All-Pairs Shortest Paths* is infeasible.
- If we accept feasible solutions:
  - We can use inadmissible macros.
  - Delete edges after some admissible macros have been introduced.

- The importance of macros is that they can be determined before the search.
  - It's called Macro learning.

Macro Problem Solving

- How to use inadmissible macros:
  - Inserting them with a weight $w(u, v)$ smaller than the optimum $\delta(u, v)$.
    - The macros will be used with higher priority.
  - Or we can restrict the search to macros only.
    - Only possible if the goal stay reachable.

- Creating inadmissible macros depends on the problem.

Macro Problem Solving

- Example:
  - We decompose the problem in subgoals.
  - For each subgoals a set of macros is defined that transform a state into the next subgoal.

Eight-Puzzle

- Actions are labeled by the direction in which the blank is moving.
- We create a table:
  - The entry in row r and in the column c contains a macro.
  - The macro is the sequence to position the tile in position r to the position c.
  - After execution the tiles in position 1 to $r-1$ remain correctly placed.

6	1	3
8	4	7
2	5	

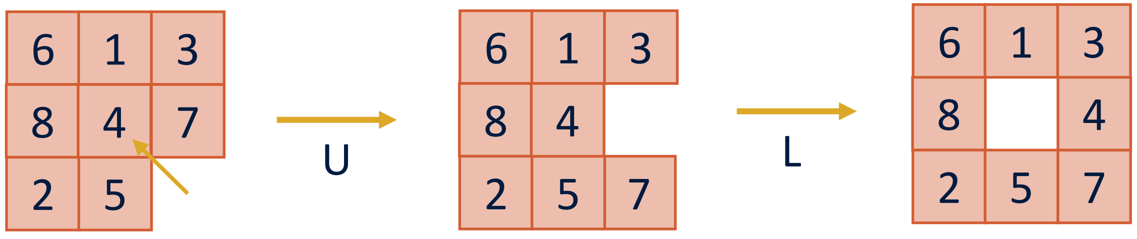
Starting state

1	2	3
8		4
7	6	5

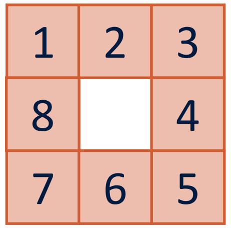
Goal state

Eight-Puzzle

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU



c=0, r=5

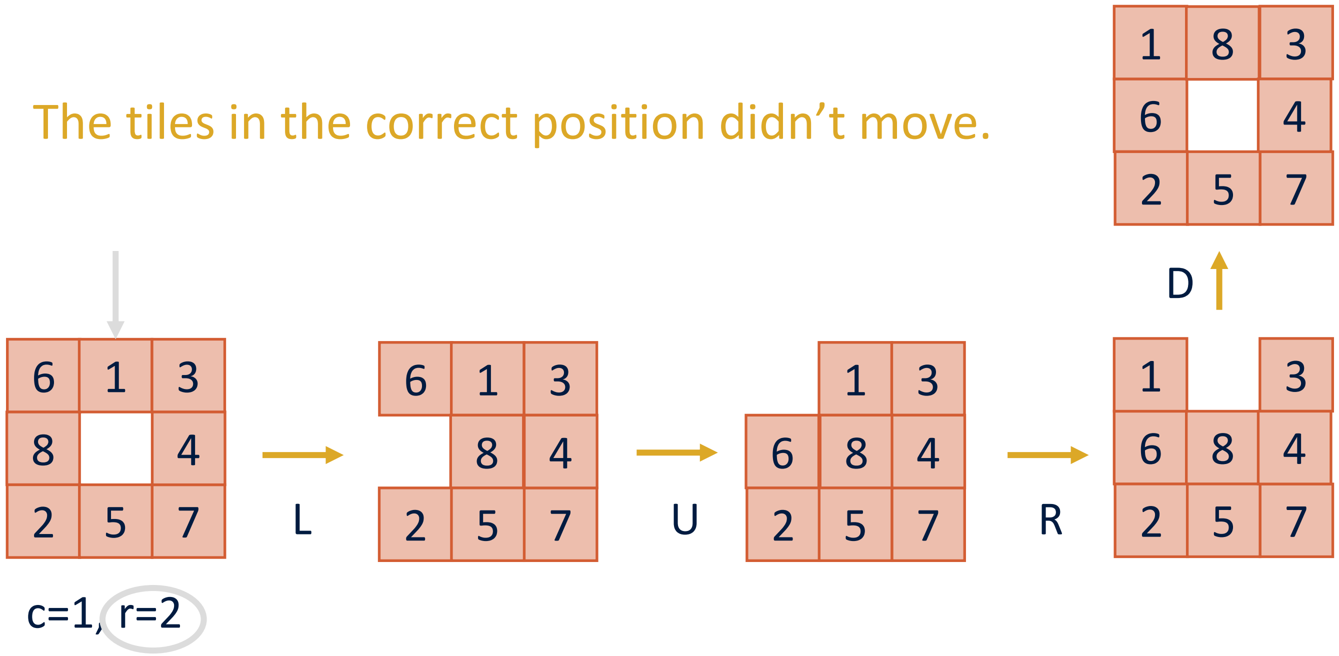


Goal state

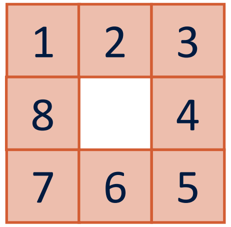
Eight-Puzzle

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU

The tiles in the correct position didn't move.



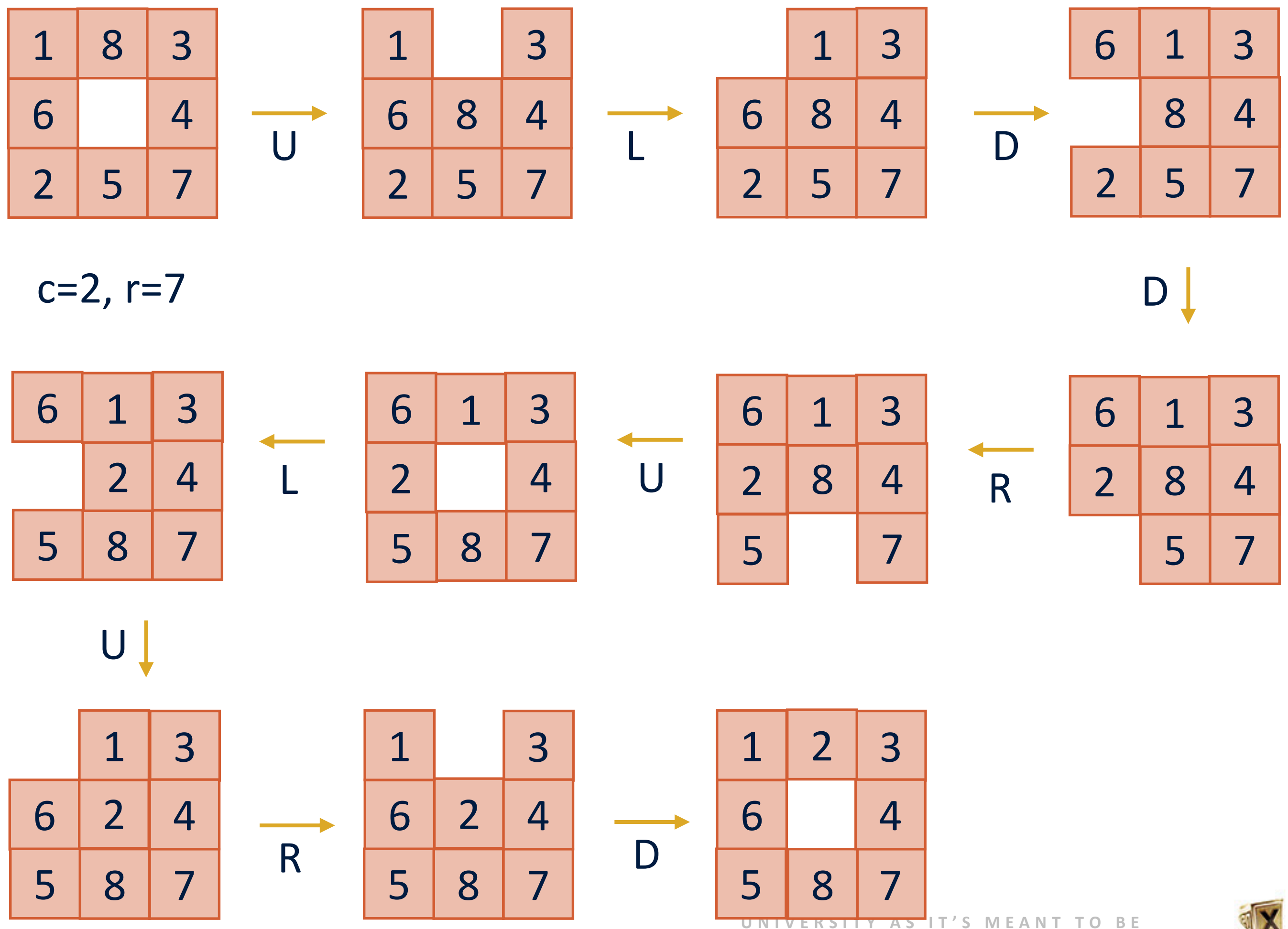
Do you see a pattern?



Goal state

Eight-Puzzle

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU



Eight-Puzzle

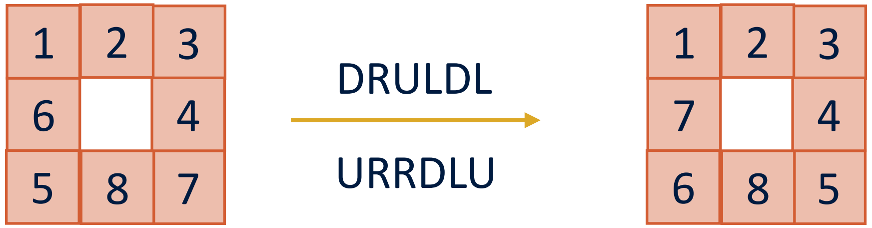
	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU

1	2	3
6		4
5	8	7

The tiles 3 and 4 are in the correct positions. We can skip it.

Eight-Puzzle

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU



c=5, r=7

The tiles in the correct positions don't move.

Eight-Puzzle

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU

c=6, r=7

It's done.

Was it optimal?

Eight-Puzzle

- In the worst-case scenario:
  - We sum the string size maxima in the columns.
  - 2+12+10+14+8+14+4=64
- The average solution length:
  - We calculate the arithmetic means.
  - 12/9+52/8+40/7+58/6+22/5+38/4+8/3=39.78
- Considering that you're using the macro table!

	0	1	2	3	4	5	6
0							
1	DR						
2	D	LURD					
3	DL	URDL LURD	URDL				
4	L	RULD LURD	RULD	LURRD LULDR			
5	UL	DRUL DLUR ULDR	RDLU RULD	RULD RDLU URDL	RDLU		
6	U	DLUR ULDR	DRUULD	DLUU RDRU LLDR	DRUL	LURRD DLURU LLDR	
7	UR	LDRU ULDR	ULDDR ULURD	LDRUL URDRU LLDR	DLUR DRUL	DRULDL URRDLU	DLUR
8	R	ULDR	LDRR UULD	LURDR ULLDR	LDRRUL	DRUL LDRU RDLU	LDRU

Eight-Puzzle

- How can we construct a macro table?
  - The most efficient way is using DFS or BFS.
  - Starting from each goal state to every other states.

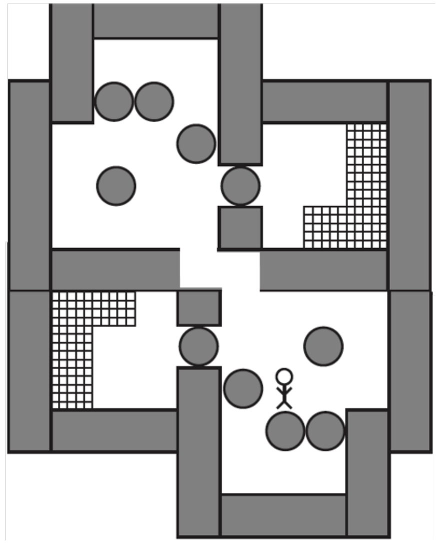
- Depending on the problem the search effort can be important.

Relevance Cuts

- Humans can navigate through large state spaces due to an ability to use meta-level reasoning.
- Meta-level strategy (reasoning) distinguish between relevant and irrelevant actions.
  - Divide a problem into several subgoals, then solve the subgoals one after the other.
- Standard search algorithm like A* always consider all possible moves available.

Relevance Cuts

- Example:
  - In mirror-symmetrical Sokoban.
  - It is obvious that each half can be solved independently.
  - Algorithm like A* will explore strategy that humans would never consider.
    - Switching back and forth between the two subproblems.



Relevance Cuts

- Relevance cuts:
  - Attempt to restrict the way the algorithm chooses the next action.
  - The idea is to prevent the program from trying all possible move sequences.
  - It introduces the notion of influence.

- Moves that don’t influence each other are called distant moves.

Relevance Cuts

- A move can be cut off:
  - If within the last m moves more than k distant moves were made.
    - This cut will discourage arbitrary switches between non-related areas of the maze.
  - Or a move that is distant with respect to the previous move, but not distant to a move in the past m moves.
    - This will not allow switches back into an area previously worked on and abandoned just briefly.

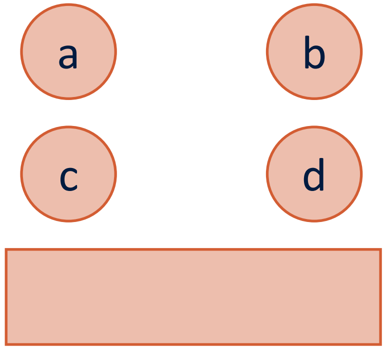
Relevance Cuts

- The definition of distant moves depends on the problem domain.

- For the Sokoban:
  - Create a measure for influence.
  - Compute a table for the influence of each square on each other.
  - The influence relation reflects the number of paths between the squares.
    - The more alternatives exists, the less influence.

Relevance Cuts

- In this example:
  - a and b influence each other less than c and d.
    - Squares on the optimal place should have a stronger influence than others.
  - a influences c more than c influences a.
    - Neighboring squares that are connected by a possible ball push are more influencing than if only the man can move between them



Relevance cuts

- Given an influence table, a move M2 is regarded as distant from a previous move M1, if its from-square influences M1’s from-square by less than some threshold, θ.

Nonadmissible State Space Pruning

- Macro problem solving prunes actions in favor of a few action sequences (called macros), which not only decreases the branching factor but also the search depth.
  - We applied it on the Eight-Puzzle where the macros bring one tile after the other into place without disturbing the tiles in the correct position.
- Relevance cuts prune actions in a state that are considered unimportant because they do not contribute to the subgoal currently pursued.
  - Actions that do not influence each other are called distant actions.
  - Relevance cuts can prune an action if more than a certain number of distant actions have been executed recently
  - We used Sokoban to illustrate relevance cuts.