CS 4260 and CS 5260
Vanderbilt University

Additional Comments on Search

This lecture assumes that you have

- Readings from Chapter 3 of ArtInt and
- Watch blind/heuristic search videos

As indicated on the Week 2 https://my.vanderbilt.edu/cs4260cs5260/schedule/

ArtInt: Poole and Mackworth, Artificial Intelligence 2E
at http://artint.info/2e/html/ArtInt2e.html

Example: What is an admissible heuristic function for the course scheduler?

Suppose

- Goal = {CS1101, CS2201, CS3250, CS3251, CS3270, CS3281, …, CS4269}
- Current state, Curr = {CS1101, CS2201}

Then

- h(Curr) = |Goal – Curr| (i.e., the size of the set difference between Goal and Curr)
- Goal – Curr = {CS3250, CS3251, CS3270, CS3281, …, CS4269}

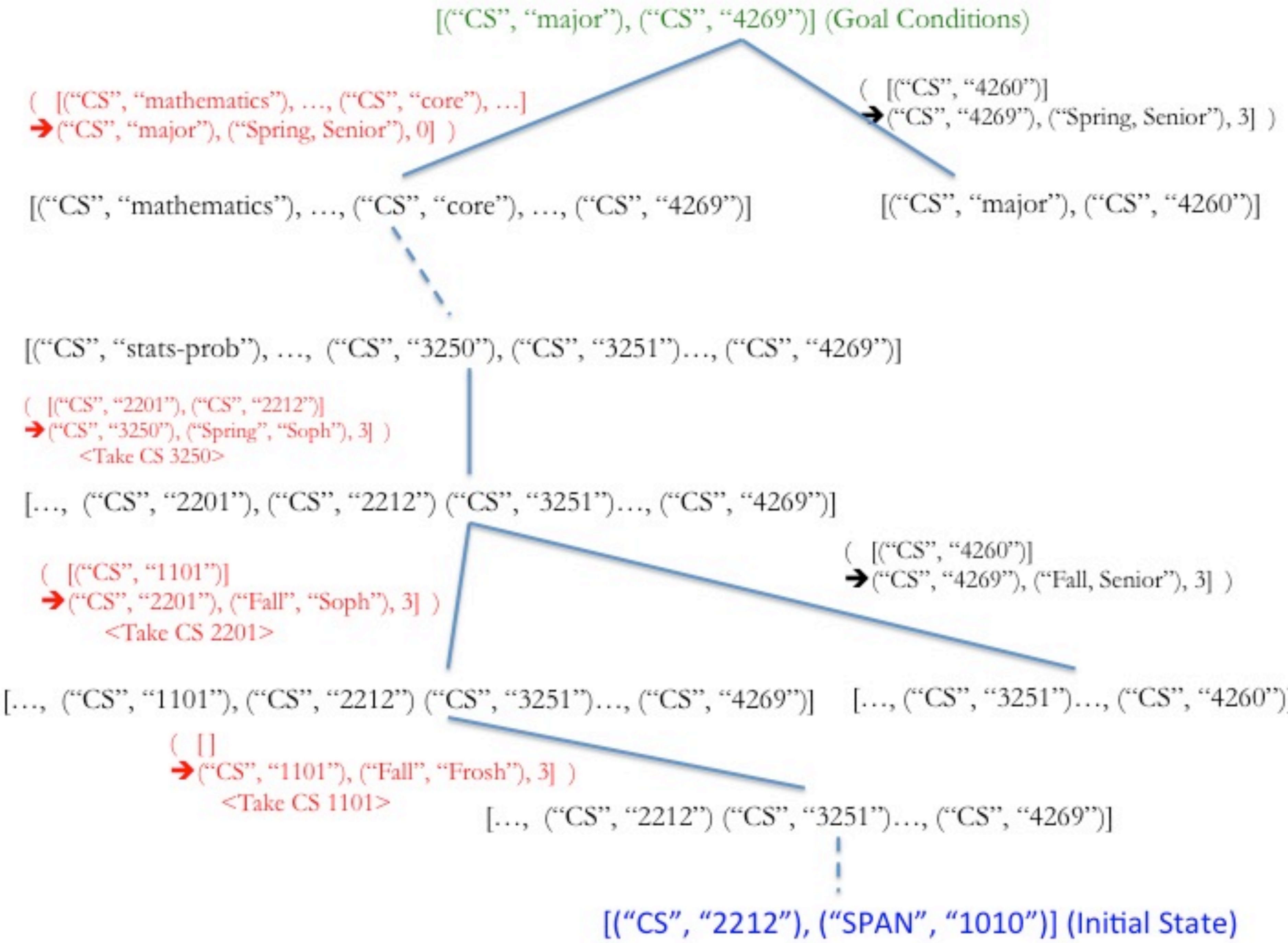
Suppose further that the Goal specification does not contain CS4260,
then h(Curr) < actual cost(Curr → Goal)

Why?

Other reasons for possible underestimate include need to repeat courses

Knowledge necessary for Goal Check, Scheduling, and reasonable user interface

492	CREO	2201	3	Fall	CREO1102
493	CS	3860	3	Spring Fall	CS2231
494	CS	3861	3	Summer Spring Fall	CS2231
495	CS	3892	3	Spring Fall	
496	CS	3890	3	Summer	
497	CS	3891	3	Fall	
498	CS	major	0	Spring Fall	CSmathematics CSsciencelab CSscienceb CSsciencec CSesintro CSliberalarts CScore CSdepth CS4959 CStechelectives CS1151 CSopenelectives CSwritingrequirement
499	CS	mathematics	0	Spring Fall	CScalculus CSstatsprobability CSmathelective
500	CS	calculus	0	Spring Fall	MATH1200 MATH1201 MATH1301 MATH2300 MATH2410, MATH1200 MATH1201 MATH1301 MATH2300 MATH2600, MATH1300 MATH1301 MATH2300 MATH24
501	CS	statsprobability	0	Spring Fall	MATH2810, MATH2820, MATH3640
502	CS	mathelective	0	Spring Fall	MATH2420, MATH2610, MATH2820, MATH3000, MATH3010, MATH3100, MATH3200, MATH3210, MATH3300, MATH3320, MATH3620, MATH3640, MATH3650, M
503	CS	sciencelab	0	Spring Fall	BSCI1100 BSCI1100L, BSCI1510 BSCI1510L, BSCI1511 BSCI1511L, CHEM1601 CHEM1601L, CHEM1602 CHEM1602L, EES1510 EES1510L, MSE1500 MSE1500L, PHY
504	CS	scienceb	0	Spring Fall	BSCI1100, BSCI1510, BSCI1511, BSCI2218, BSCI2219, CHEM1601, CHEM1602, EES1510, MSE1500, PHYS1601, PHYS1602
505	CS	sciencec	0	Spring Fall	BSCI1100, BSCI1510, BSCI1511, BSCI2218, BSCI2219, CHEM1601, CHEM1602, EES1510, MSE1500, PHYS1601, PHYS1602
506	CS	liberalarts	0	Spring Fall	CSliberalhum CSliberalsoc CSliberalother
507	CS	liberalhum	0	Spring Fall	HIST2700 ENGL3896, ENGL1250W EUS2203
508	CS	liberalsoc	0	Spring Fall	PSY1200 SOC3702, SOC3321 ANTH4154
509	CS	liberalother	0	Spring Fall	ARTS1102 ARTS2101, ARTS1102 ARTS2102
510	CS	esintro	0	Spring Fall	ES1401 ES1402 ES1403
511	CS	core	0	Spring Fall	CS1101 CS2201 CS3251 CS3270 EECE2116 EECE2116L CS2231 CS3281 CS2212 CS3250
512	CS	depth	0	Spring Fall	CSdepthproject CSdepthothera CSdepthotherb CSdepthotherc
513	CS	depthproject	0	Spring Fall	CS3259, CS3892, CS4269, CS4279, CS4287
514	CS	depthothera	0	Spring Fall	CS3259, CS3282, CS3860, CS3861, CS3892, CS4260, CS4278, CS4285, CS4287, CS4959, CS3252, CS3265, CS3274, CS4269, CS4279, CS4283, CS3890, CS428
515	CS	depthotherb	0	Spring Fall	CS3259, CS3282, CS3860, CS3861, CS3892, CS4260, CS4278, CS4285, CS4287, CS4959, CS3252, CS3265, CS3274, CS4269, CS4279, CS4283, CS3890, CS428
516	CS	depthotherc	0	Spring Fall	CS3259, CS3282, CS3860, CS3861, CS3892, CS4260, CS4278, CS4285, CS4287, CS4959, CS3252, CS3265, CS3274, CS4269, CS4279, CS4283, CS3890, CS428
517	CS	writingrequirement	0	Spring Fall	AADS3104W, AADS3204W, AADS4228W, AMER1002W, ANTH1201W, ANTH2113W, ANTH2220W, ANTH3150W, ANTH3243W, ANTH3622W, ANTH2242W, ANTH1
518	CS	techelectives	0	Spring Fall	CStechelectives1 CStechelectives2
519	CS	techelectives1	0	Spring Fall	BME2100, BME3000, BME3100, BME3110, BME3200, BME3300, BME3860, BME3861, BME3890, BME4100, BME4200, BME4420, BME4600, BME4900W, BME4950, B
520	CS	techelectives2	0	Spring Fall	BME2100, BME3000, BME3100, BME3110, BME3200, BME3300, BME3860, BME3861, BME3890, BME4100, BME4200, BME4420, BME4600, BME4900W, BME4950, B
521	CS	openelectives	0	Spring Fall	CSopen1 CSopen2 CSopen3 CSopen4 CSopen5 CSopen6
522	CS	open1	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
523	CS	open2	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
524	CS	open3	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
525	CS	open4	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
526	CS	open5	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
527	CS	open6	0	Spring Fall	BASS1000, BSSN1000, CLAR1000, CLLO1000, COMP1000, FLUT1000, GTR1000, HARP1000, HORN1000, MUSO1000, OBOE1000, PERC1000, PIAN1000, SAX1000,
528	CS	4959	1	Fall	CS3281
529	CS	1101	3	Summer Spring Fall	
530	CS	1103	3	Summer Spring Fall	
531	CS	1151	3	Spring Fall	
532	CS	2201	3	Spring Fall	CS1101


[("CS", "major"), ("CS", "4269")] (Goal Conditions)
([("CS", "mathematics"), …, ("CS", "core"), …]
➔("CS", "major"), ("Spring, Senior"), 0])
([("CS", "4260")]
➔("CS", "4269"), ("Spring, Senior"), 3])
[("CS", "mathematics"), …, ("CS", "core"), …, ("CS", "4269")]
[("CS", "major"), ("CS", "4260")]
[("CS", "stats-prob"), …, ("CS", "3250"), ("CS", "3251")…, ("CS", "4269")]
([("CS", "2201"), ("CS", "2212")]
➔("CS", "3250"), ("Spring", "Soph"), 3])
<Take CS 3250>
[…, ("CS", "2201"), ("CS", "2212") ("CS", "3251")…, ("CS", "4269")]
([("CS", "1101")]
➔("CS", "2201"), ("Fall", "Soph"), 3])
<Take CS 2201>
([("CS", "4260")]
➔("CS", "4269"), ("Fall, Senior"), 3])
[…, ("CS", "1101"), ("CS", "2212") ("CS", "3251")…, ("CS", "4269")]
[…, ("CS", "3251")…, ("CS", "4260")]
([]
➔("CS", "1101"), ("Fall", "Frosh"), 3])
<Take CS 1101>
[…, ("CS", "2212") ("CS", "3251")…, ("CS", "4269")]
[("CS", "2212"), ("SPAN", "1010")] (Initial State)

State Space Search Application: Rubix Cube Solver

Set of states: some configuration of the Rubix cube

Start state: a given configuration of the Rubix cube given to the Rubix Cube Solver

Goal state: configuration of Rubix Cube in which each face is only of a single color

Action function: A series of rotations that solve a particular subproblem, determined by which subcubes should be "switched"

I think a rubiks cube solver is a classic example in which artificial intelligence can utilize a dumb algorithm layer to offload work. One can just as easily define the action functions to be a single rotation. However, study of the problem reveals that there can be a set of predefined algorithms for switching subcubes in particular patterns. Thus, the determination of which ones to switch and whether that can get you closer to the solution requires a conscious choice.

A PROGRAM THAT LEARNS TO SOLVE RUBIK'S CUBE, Richard E. Korf
https://www.aaai.org/Papers/AAAI/1982/AAAI82-039.pdf

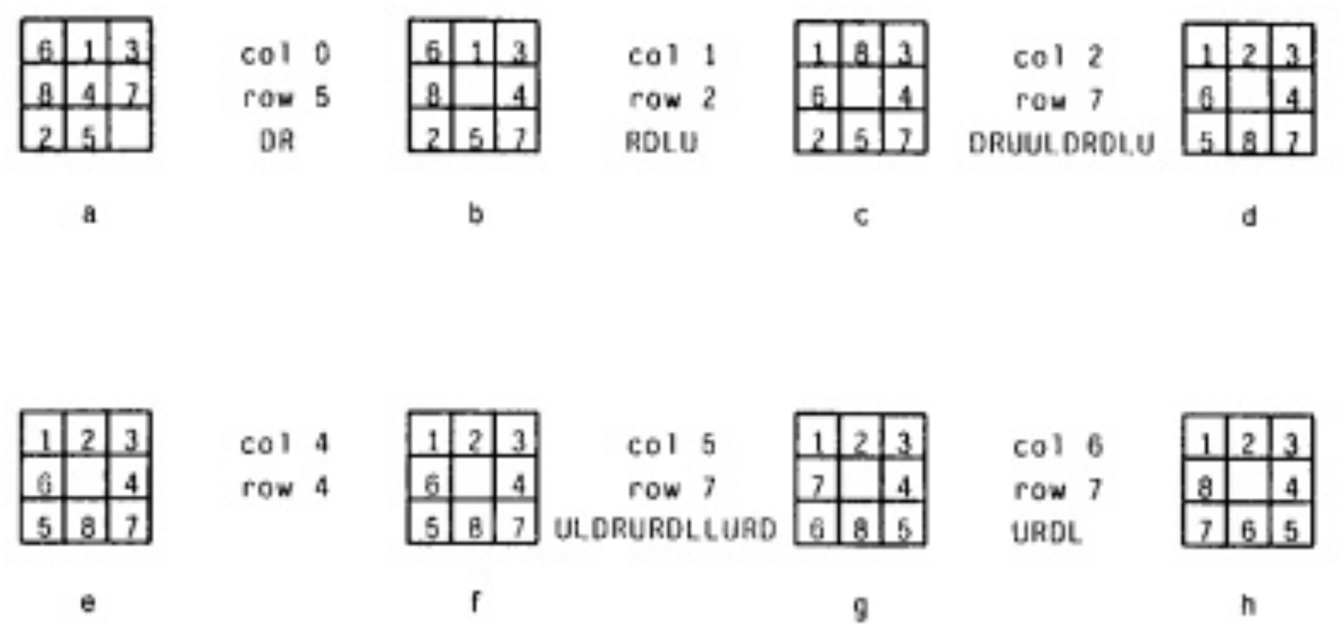
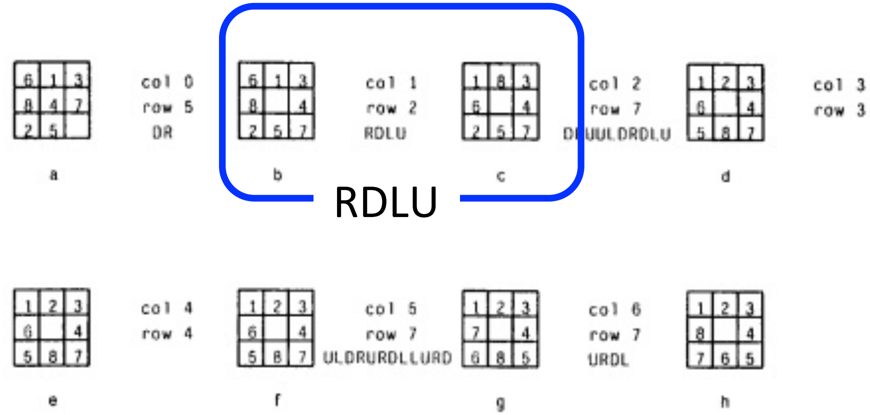
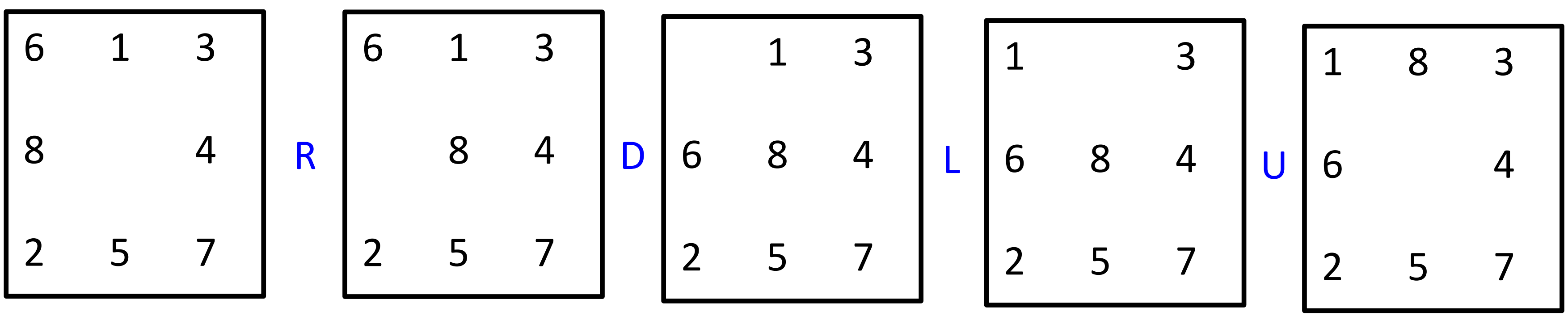


FIG. 7. Example of solution of the Eight Puzzle by the Macro Problem Solver.

Macro-Operators: A Weak Method for Learning, by Richard Korf, From https://www.sciencedirect.com/science/article/pii/0004370285900128



Macro-Operators: A Weak Method for Learning, by Richard Korf, From https://www.sciencedirect.com/science/article/pii/0004370285900128



RDLU is a macro that places 1 in is goal position and returns all Previous "subgoals" (i.e., blank in this case) to their goal position

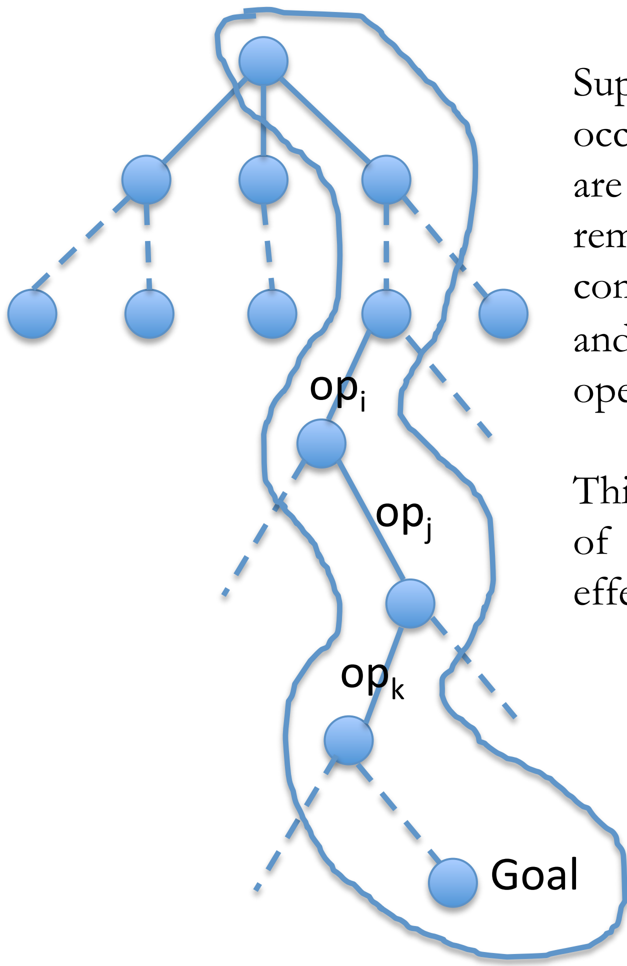
Macro-Operators: A Weak Method for Learning, by Richard Korf, From https://www.sciencedirect.com/science/article/pii/0004370285900128

TABLE 1. Macro table for the Eight Puzzle. The total number of non-identity macros is 35. The average case solution length is 39.78 moves

	Tiles						
Positions	0	1	2	3	4	5	6
0							
1	UL						
2	U	RDLU					
3	UR	DLURRDLU	DLUR				
4	R	LDRURDLU	LDRU	RDLLURDRUL			
5	DR	ULDRURDLDRUL	LURDLDRU	LDRULURDDLUR	LURD		
6	D	URDLDRUL	ULDDRU	URDDLULDRRUL	ULDR	RDLLUURDLDRRUL	
7	DL	RULDDRUL	DRUULDRDLU	RULDRDLULDRRUL	URDLULDR	ULDRURDLLURD	URDL
8	L	DRUL	RULLDDRU	RDLULDRRUL	RULLDR	ULDRRULDLURD	RULD

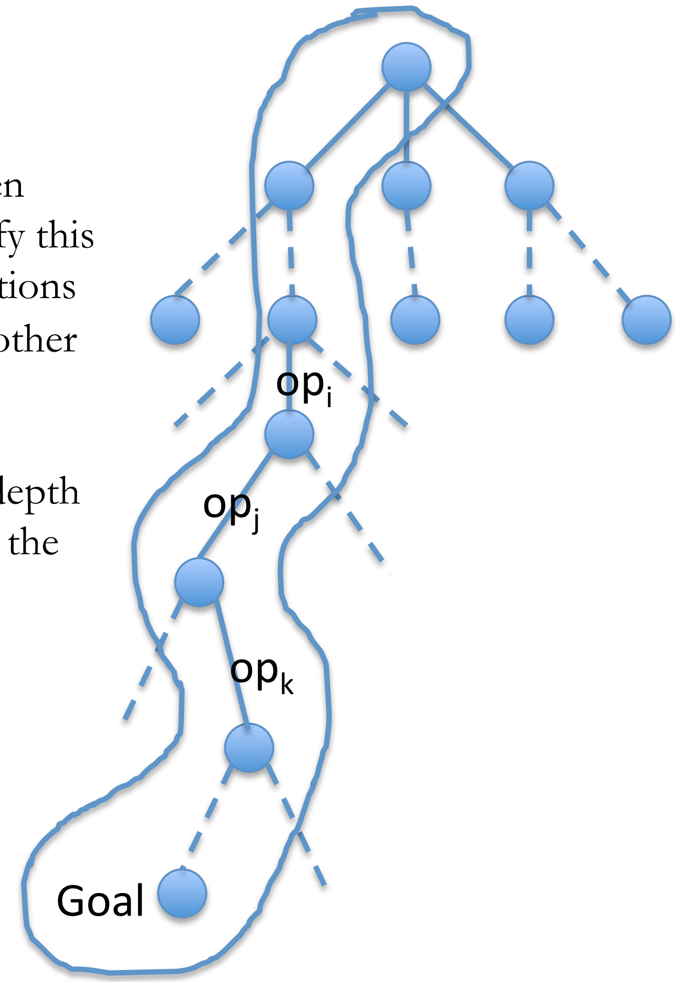
Why are composite (aka macro) operators useful?

Operators that frequently occur "back-to-back" may be useful to remember as a package



Suppose that op_i, op_j, and op_k occur frequently in plans that are found through search. Then remember op_i; op_j; op_k, identify this composite operators preconditions and effects, and treat like any other operator during search

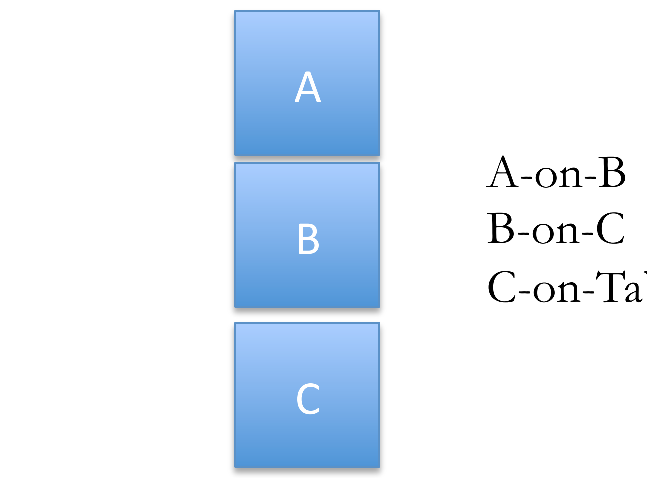
This can reduce the effective depth of search, but it also increases the effective breadth of search



Why are composite (aka macro) operators useful?

More interesting reason: macros can bridge places in the search where the heuristic is misleading

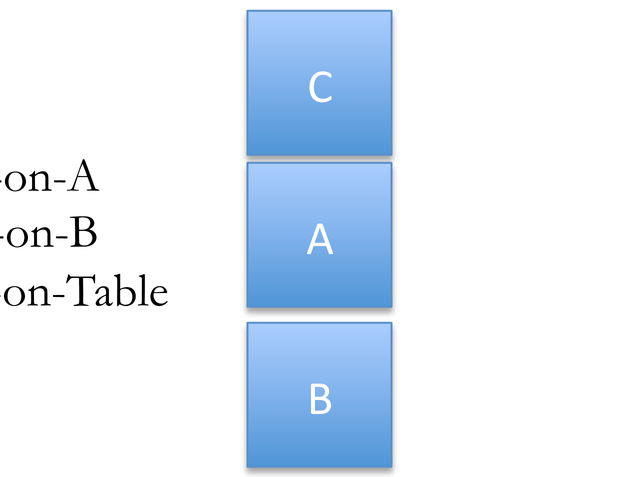
Consider this situation



A-on-B
B-on-C
C-on-Table

Initial State

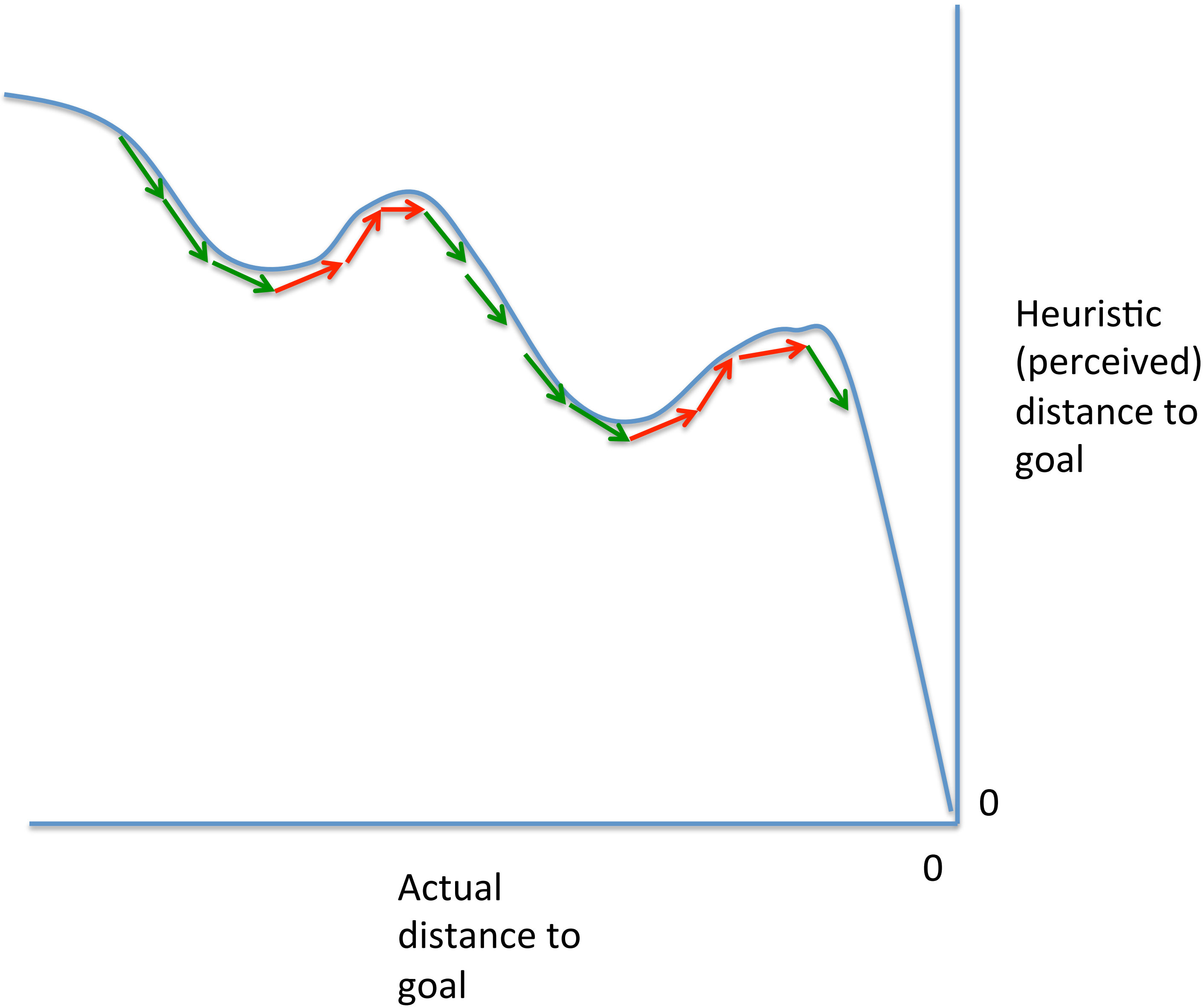
C-on-A
A-on-B
B-on-Table

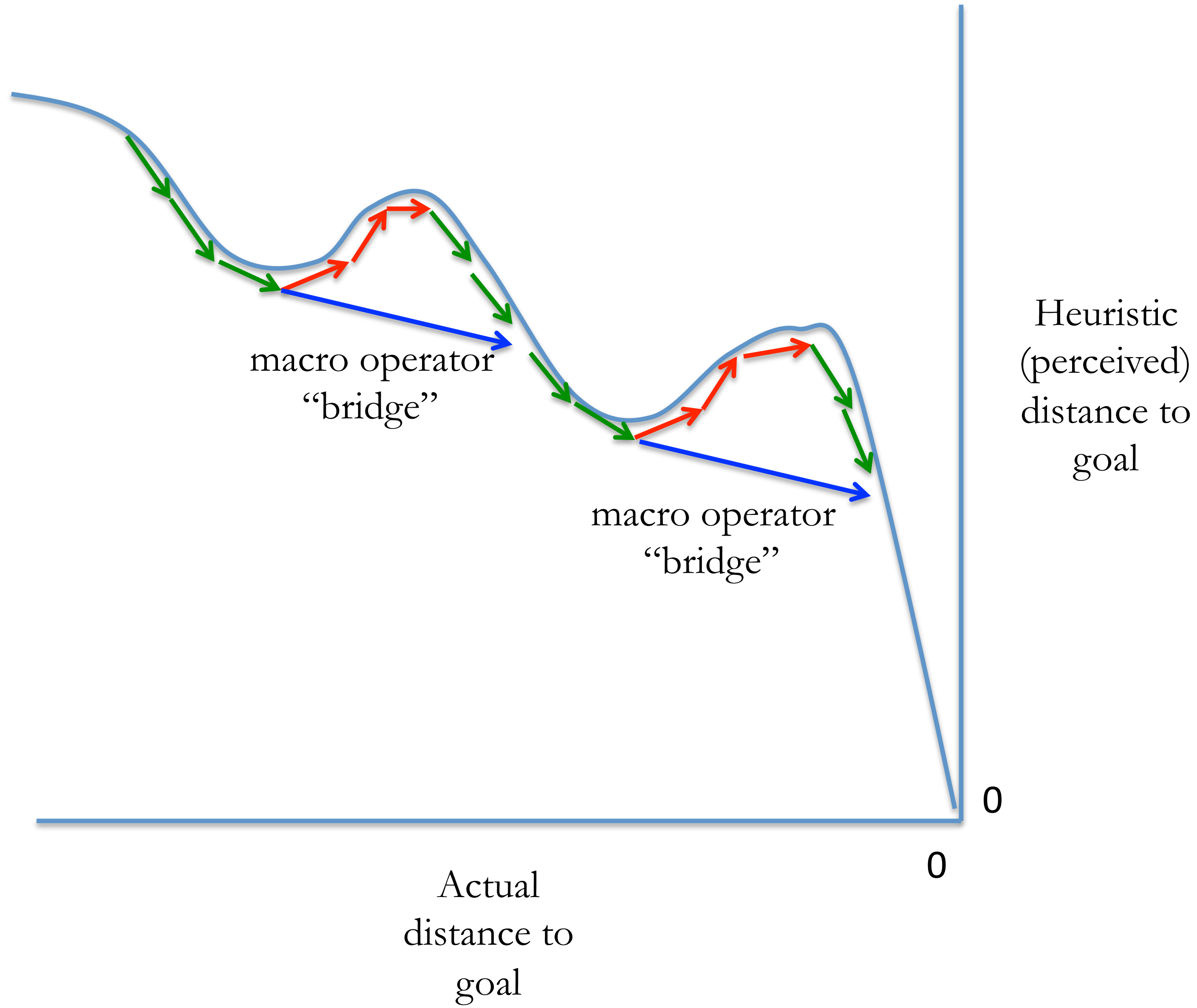


Goal State

Use forward search, with
heuristic that counts the number
of unachieved subgoals
so, h(Initial State) = 2

but it is necessary to use the unstack operator to remove A from B
to eventually achieve the final goal. This resulting intermediate state has an h value of 3


Heuristic
(perceived)
distance to
goal
0
0
Actual
distance to
goal


macro operator
“bridge”
macro operator
“bridge”
Heuristic
(perceived)
distance to
goal
0
0
Actual
distance to
goal

State Space Search - planning a road trip
Get A Good Software Engineering Job
Deciding which items to pack for a trip
State Space Search: Tic Tac Toe
Putting an outfit together
A robot that needs to navigate a room or building
Groceries State Space Search
Potential Application - Meal Planning
Analysis of Song Choice
Searching for a File
Rider-matching
Sudoku formulated as State Space Search
State Space Search Application: Rubix Cube Solver
Career Coach
Food Planning
Construction of a generic physical product
Travel salesman problem
Finding a route (from Rand to Commons)
Road trip
Sudoku as a State Space Search
College Scheduling
Path-finding

State Space Search for Chess
Booking Cheapest Flights
People that need to cross a river (scheduling under constraints)
Game of chess
Football AI app
Tic Tac Toe
State Space Search in a Parser
Designing new chemical compounds
Personal Fitness Tracker
Winning a game of Catan
Packing Warehouse Robots
Dog shelter
Maze problem
Drone planning
Navigation State Space Search Application
Fish-feeding Robot - State Space Search
Model Checking in Formal Verification
Rand Meal Robot
Elevator control program
Following a Cake Baking Recipe
Playing your turn in Uno
Settlers of Catan Initial Piece Placement

“With the dissemination of AI into everyday life, recent years have shown many new applications of heuristic search algorithms to novel domains. These domains include feature selection for clustering algorithms (Marino and Lelis 2015), ˜ anomaly detection for cyber-security (Mirsky et al. 2015), finding error-correction codes (Palombo et al. 2015), a Kivalike domain for multi-agent pathfinding (Cohen, Uras, and Koenig 2015), Maximum a Posteriori Estimation in Probabilistic Programs (Tolpin and Wood 2015), an AI player of a commercial video game which accounts for the player’s enjoyment (Churchill and Buro 2015), and **automated discovery of chemical compounds (Heifets and Jurisica 2012).** These applications of heuristic search theory and algorithms provides yet another demonstration of the impact of researching heuristic search methods.”

What's Hot in Heuristic Search? - Association for the Advancement of ...
https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12234/12283
by R Stern - 2016 - Cited by 1 - Related articles
involve searching in large state spaces. Therefore, most AI algorithms and applications include heuristic search algo- rithms. We observed in the past years an ...