SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence

2018/2019 Prof. Daniele Nardi

Exercises 2:

Search in the State Space*

Francesco Riccio
email: riccio@diag.uniroma1.it

*The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello, Prof. Daniele Nardi, Dott. Fabio Previtali and Andrea Vanzo.

Exam 17/9/2012

The following numbers are to be put in ascending order

4, 1, 3, 2

- At each step, while performing the reordering, it is possible to exchange the number in the i-*th* position, with the number in the j-*th* position
- Assume the **cost** of each move is $|j - i| + 1$
- Consider as the heuristic function **h(n)** the number of **misplaced numbers** with respect to the final position

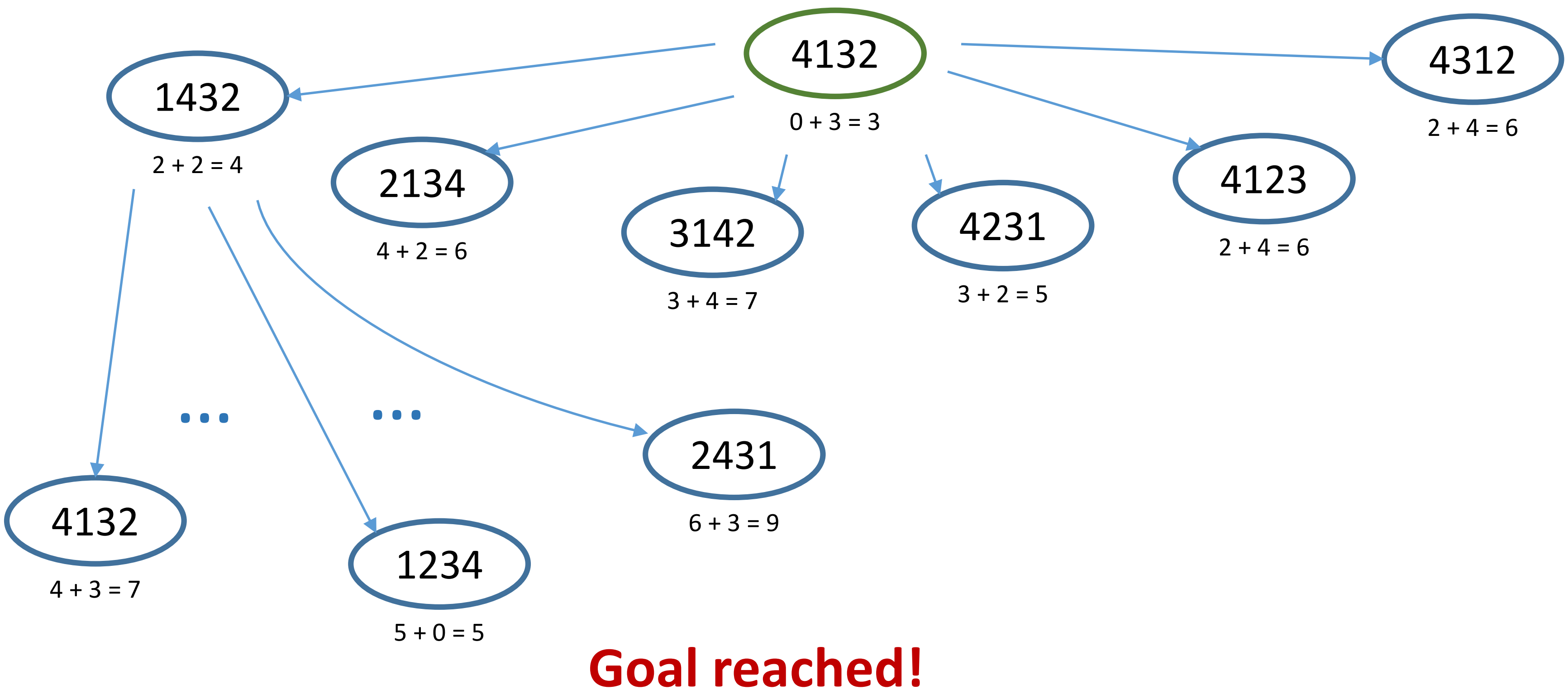
Is h(n) an **admissible** heuristic?

The heuristic function h(n) is admissible, as it assumes that each number can be put into its place with a cost of one, which underestimates the real cost, which is at least 2

Exam 17/9/2012

Goal state: 1,2,3,4

g(s) = |i - j| +1, h(s) = number of misplaced numbers



Exam 11/1/2013

Given two admissible heuristic functions $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$ for a problem, neither one dominating the other one

1. How a new heuristic function $\boldsymbol{h_3}$ can be constructed from h_1 and h_2 that is still admissible and **dominates** both of them?

h_3 can be defined as $\boldsymbol{h_3(n) = max\big(h_1(n), h_2(n)\big),\ \forall n \in N}$

h_3 is admissible as both h_1 and h_2 are

2. Why h_3 is to be preferred over h_1 and h_2?

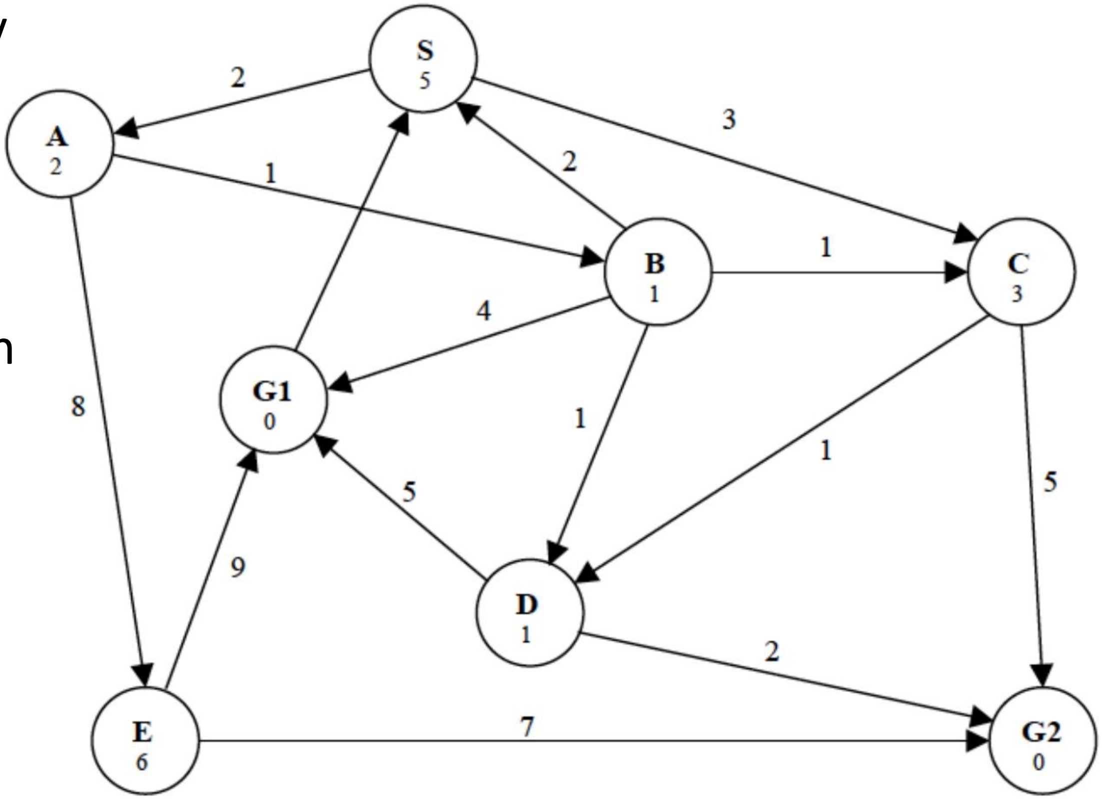
It is preferable over h_1 and h_2 because dominating heuristics are more discrimintating. Hence, they guide the search towards the goal state more faster

Exam 16/1/2015

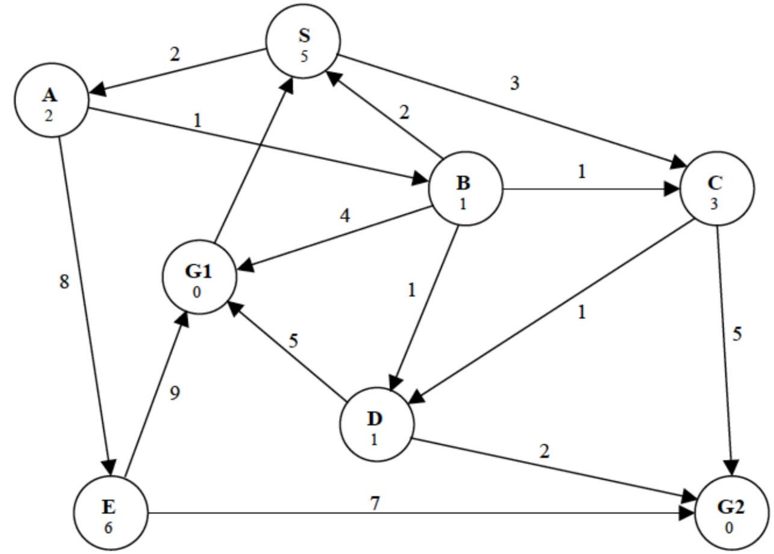
Consider the search space below, where **S** is the start node and **G1** and **G2** satisfy the goal states. Arcs are labelled with the **cost** of traversing them and the **estimated cost** to a goal is reported inside nodes.

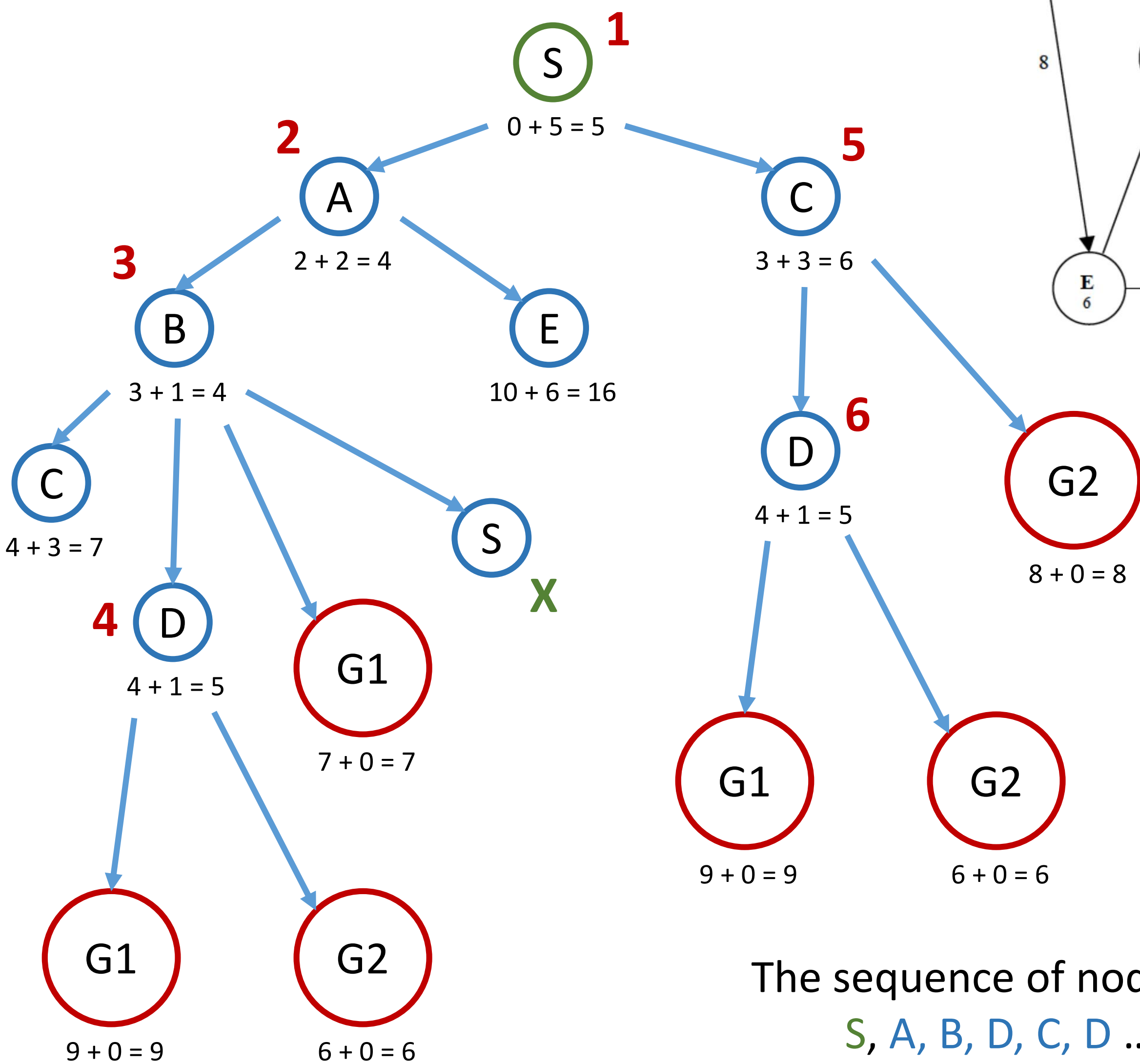
1. Draw the **tree** generated by the algorithm **A***

2. **List - in order - all the states popped of the OPEN list**. That is to say: mark with increasing natural number the nodes of the tree in the order of their expansion. When all values are equal, nodes are expanded in alphabetical order



Exam 16/1/2015



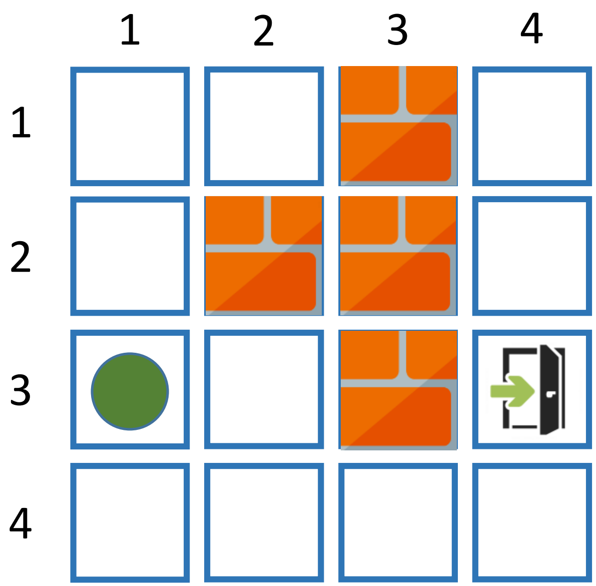


Continue…

The sequence of nodes expanded by A* is
S, A, B, D, C, D …

Exam 17/6/2015

An agent is posed at the entrance of the following labyrinth and, it has to traverse it to reach the exit . The symbol represents a wall



The **cost** for going **forward** or **up** is **1**, while for going **down** or **on diagonals** is **2**

The **state space** is the set of possible positions, that can be represented as a pair $\langle i, j \rangle$, with $\mathbf{0 < i < 5}$ and $\mathbf{0 < j < 5}$ and $\langle \boldsymbol{i}, \boldsymbol{j} \rangle \neq$

The **initial** state is in $\langle \mathbf{3}, \mathbf{1} \rangle$ while the **goal** state is in $\langle \mathbf{3}, \mathbf{4} \rangle$

At each step, the agent can move in every direction to one of the adjacent cells. It can perform an horizontal, vertical and diagonal move and, it can only advance from left to right, i.e. it cannot go from $\langle i, j \rangle$ to $\langle i, j-1 \rangle$, $\langle i-1, j-1 \rangle$ not $\langle i+1, j-1 \rangle$. Of course, the agent cannot traverse walls nor move out of the grid

Exam 17/6/2015

Operators

Name	Meaning	Effect	Cost
$up(\langle i,j\rangle)$	Go up	$\langle i+1,j\rangle$	1
$d-up(\langle i,j\rangle)$	Go diagonal up	$\langle i+1,j+1\rangle$	2
$forward(\langle i,j\rangle)$	Go forward	$\langle i,j+1\rangle$	1
$down(\langle i,j\rangle)$	Go down	$\langle i-1,j\rangle$	2
$d-down(\langle i,j\rangle)$	Go diagonal down	$\langle i-1,j+1\rangle$	2

Exam 17/6/2015

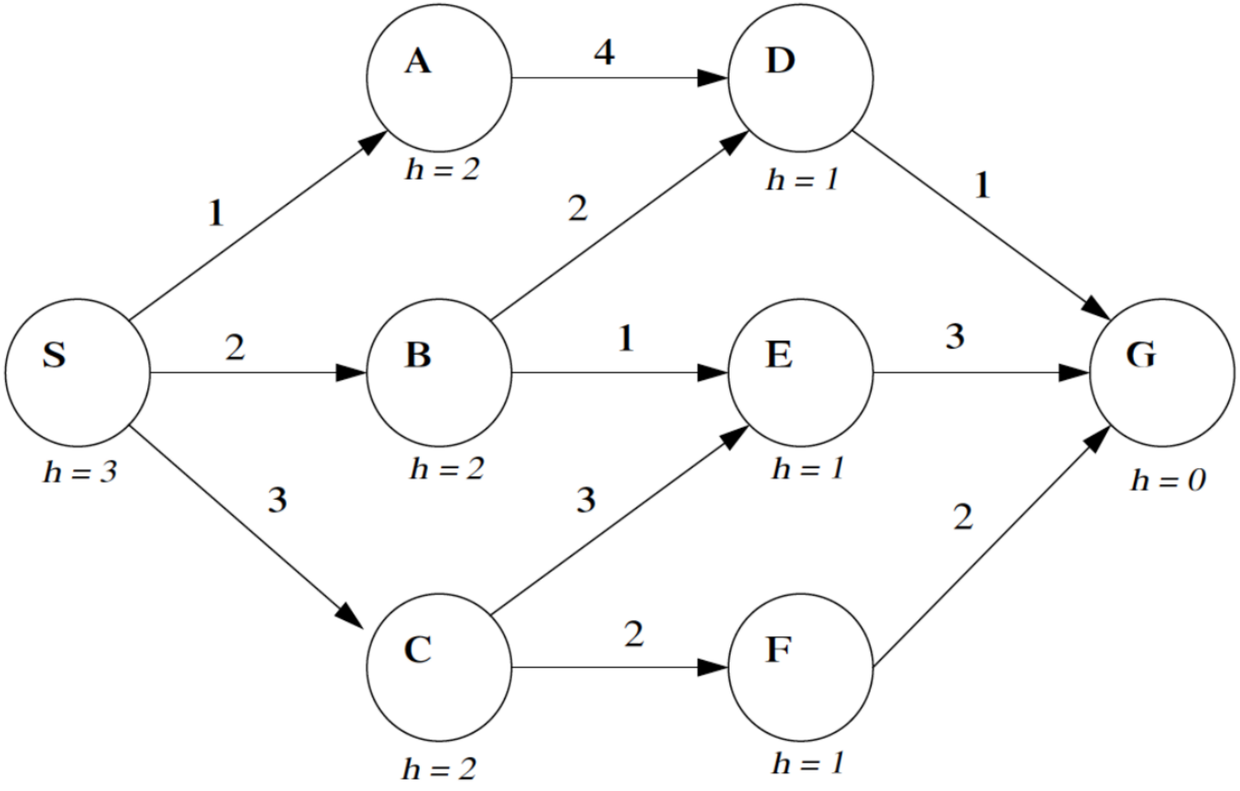
1. Is the **Manhattan distance** an appropriate **heuristic** function? Is it admissible?

2. List the expanded nodes in the **order of expansion**

3. Use **A*** with the above heuristic function to find a solution. **List all the generated nodes** with the order of generation and the values for g, h and f. **When more than one node has the same minimal value for f expand the most recently generated one**

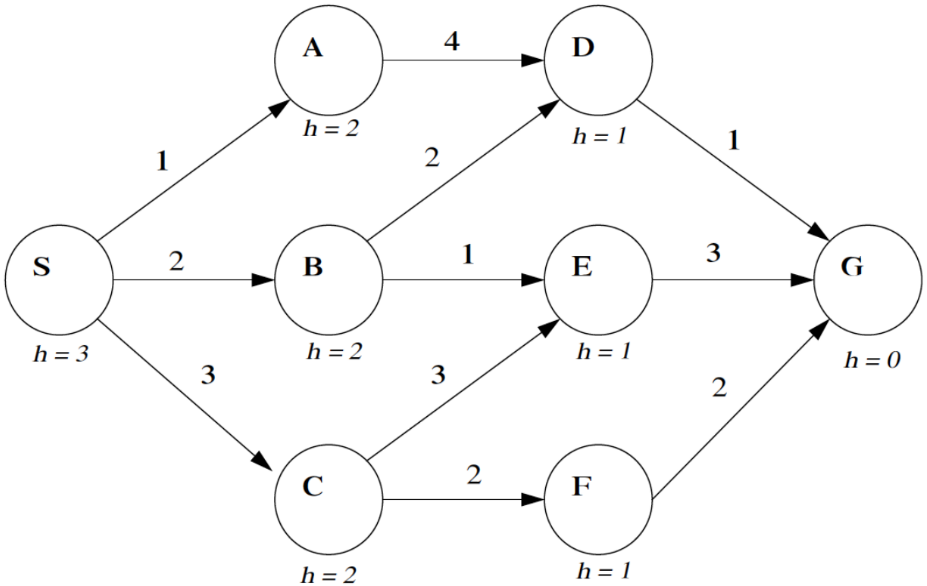
Exam 13/2/2015

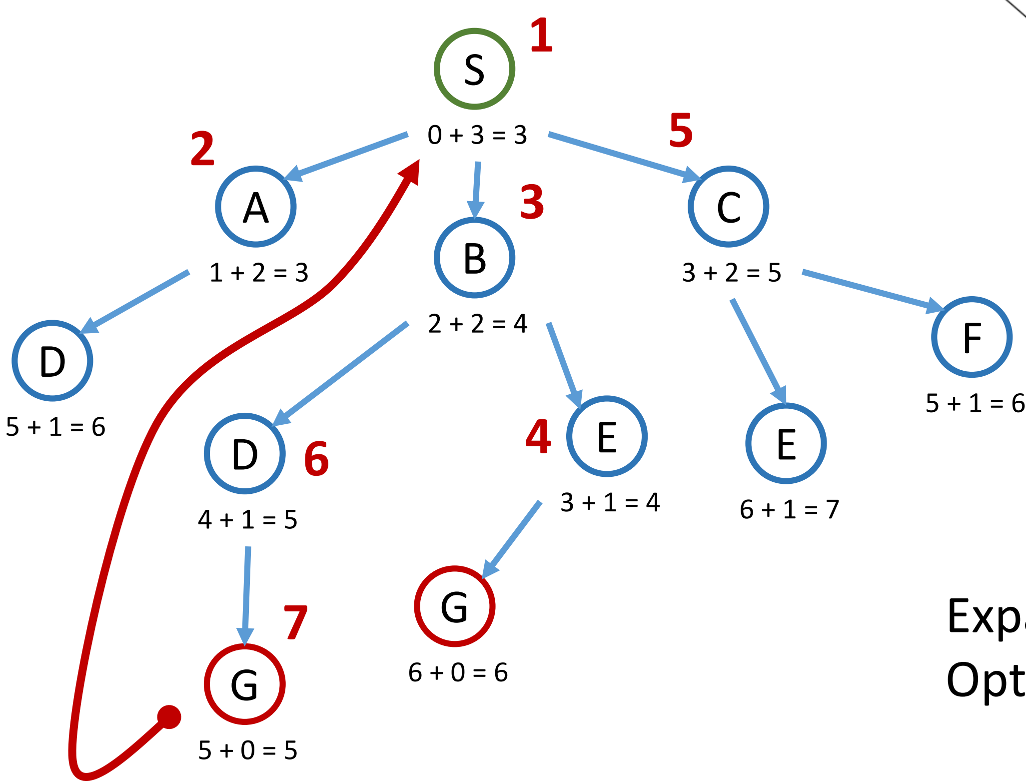
Consider the search problem represented in the following **graph**. It has start state **S** and goal state **G**. Transition **costs** are shown as numbers on the arrows. **Heuristic** values are shown below each state.



Exam 13/2/2015

1. Draw the A* search tree
2. Mark the expansion order
3. Show the optimal solution path



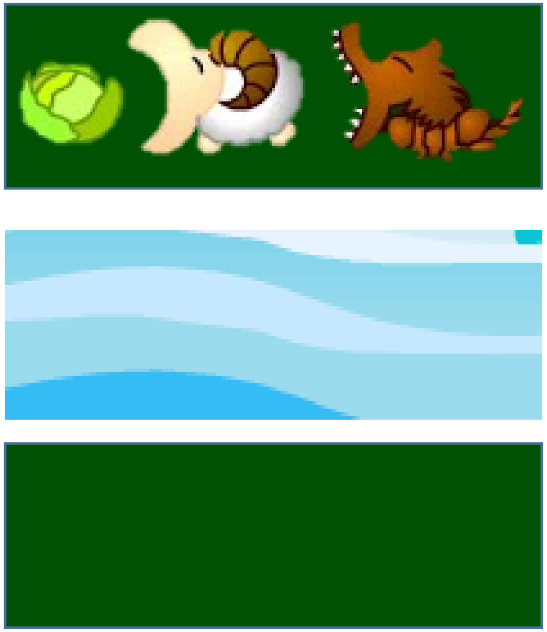


Expansion order: S, A, B, E, C, D, G

Optimal solution: S, B, D, G

Example: Crossing the river

A man has a **wolf**, a **sheep** and a **cabbage**. He is on a river bench with a **boat**, whose maximum **load** for a single trip is the man **plus one of his 3 goods**. The man wants to cross the river with his goods, but he must avoid that - when he is far away - **the wolf eats the sheep** and that, **the sheep eats the cabbage**. How can the man reach is goal?



1. Characterize the **state space**
2. Specify the **operators**
3. Find a minimal sequence of moves to **solve the problem**
4. Find a good **heuristics** to be used by A∗
5. Draw the **search tree generated by A∗**. For each node indicate: the number (state), *f*, *g*, and *h* and an integer indicating the expansion order

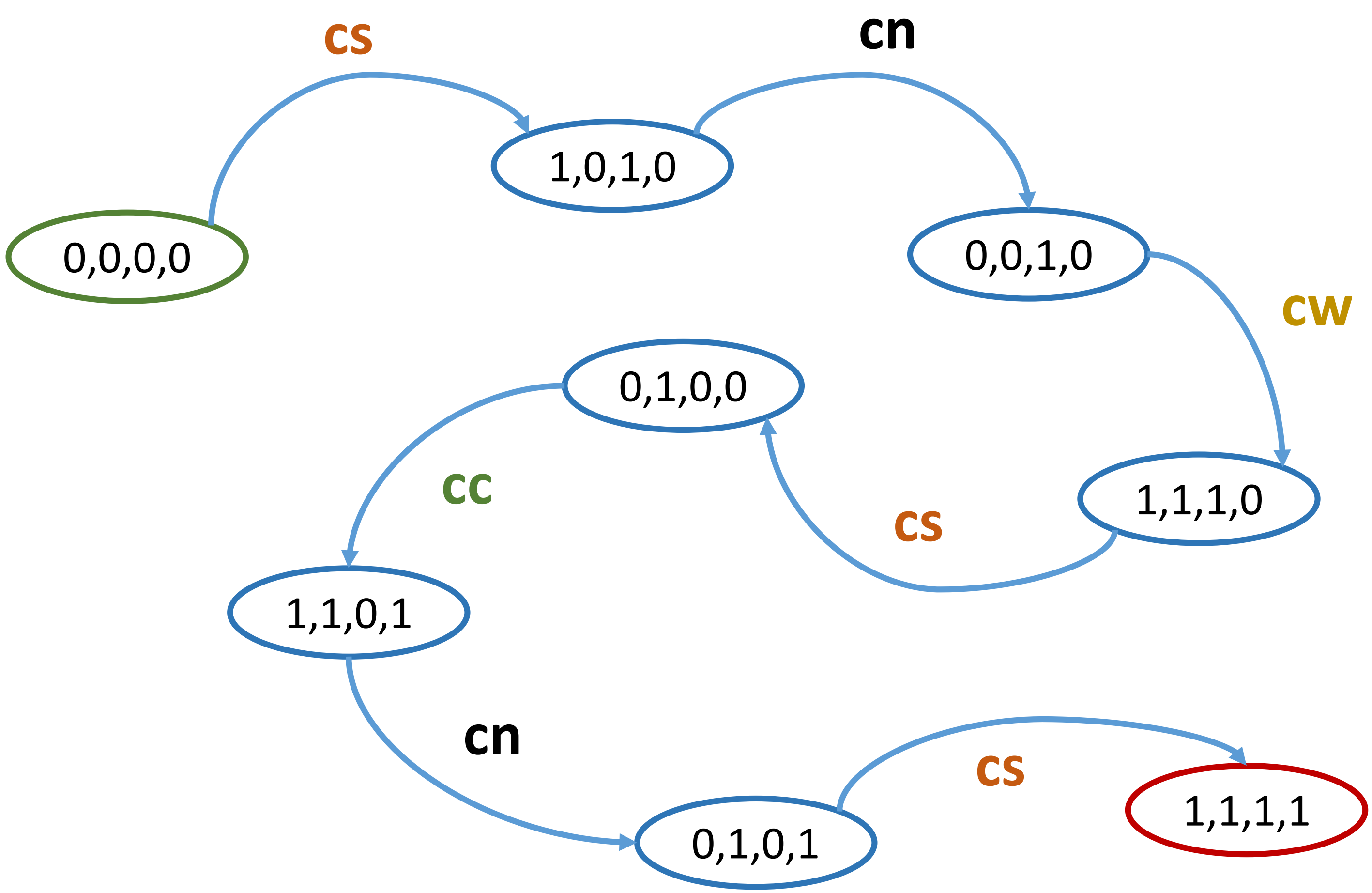
State space

A man has a **wolf**, a **sheep** and a **cabbage**. He is on a river bench with a **boat**, whose maximum **load** for a single trip is the man **plus one of his 3 goods**. The man wants to cross the river with his goods, but he must avoid that - when he is far away - **the wolf eats the sheep** and that, **the sheep eats the cabbage**. How can the man reach is goal?

1. Characterize the **state space**

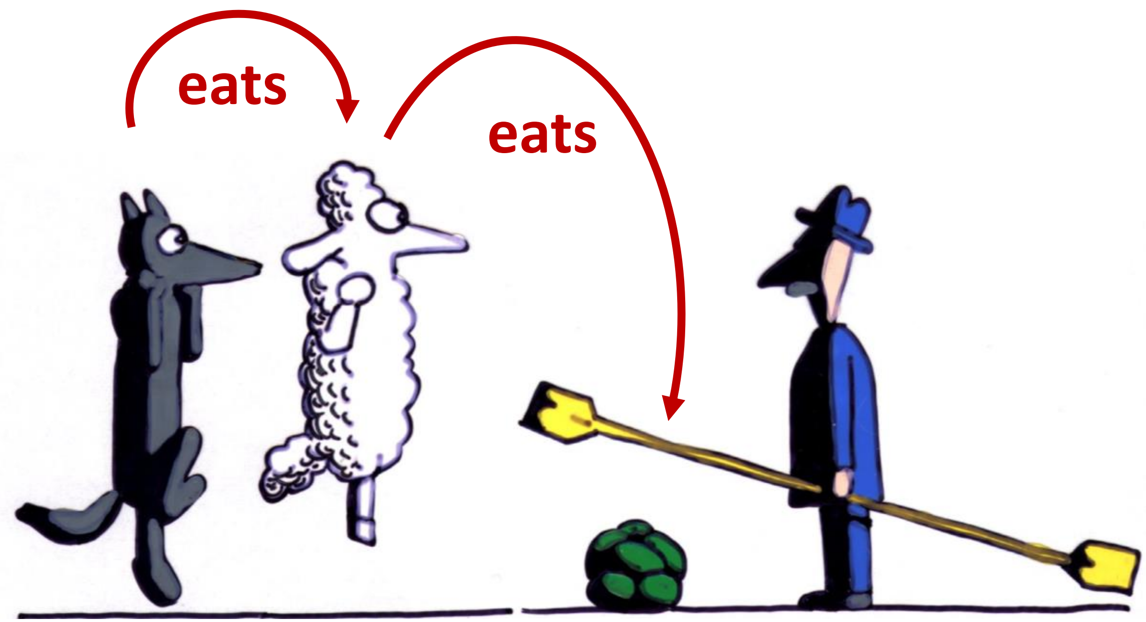
- Let $S = D \text{ x } D \text{ x } D \text{ x } D$ where $D = \{0,1\}$ and *0* and *1* are represent the river benches
- $\langle M, W, S, C \rangle \in S$ represents the position of the man, the wolf, the sheep and the cabbage
- **Initial state**: $\langle 0,0,0,0 \rangle$, and **Goal state**: $\langle 1,1,1,1 \rangle$

Possible solution



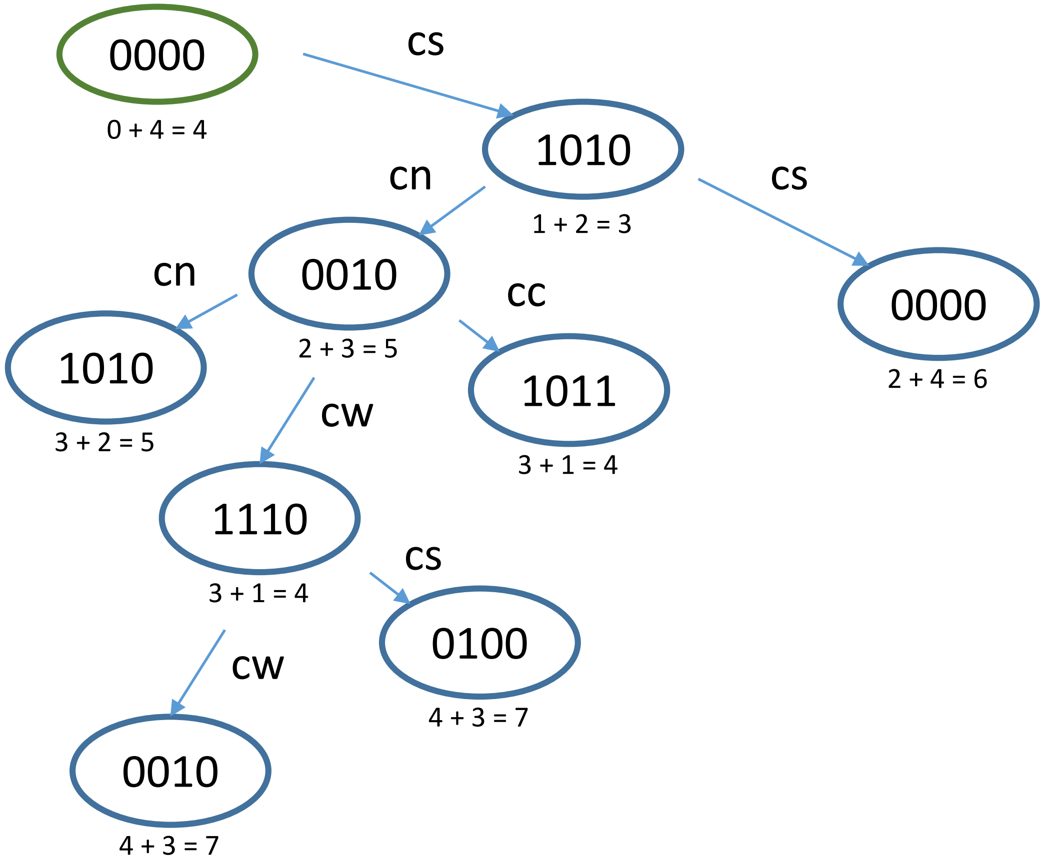
Operators

Name	Conditions	from State	to State
carryNothing (cn)	$W \neq S, S \neq C$	$\langle M, W, S, C \rangle$	$\langle \bar{M}, W, S, C \rangle$
carryWolf (cw)	$M = W, S \neq C$	$\langle M, W, S, C \rangle$	$\langle \bar{M}, \bar{W}, S, C \rangle$
carrySheep(cs)	$M = S$	$\langle M, W, S, C \rangle$	$\langle \bar{M}, W, \bar{S}, C \rangle$
carryCabbage(cc)	$M = C, W \neq S$	$\langle M, W, S, C \rangle$	$\langle \bar{M}, W, S, \bar{C} \rangle$



Heuristics & A*

cost = 1 for each trip, $h(s) = |4 - M - W - S - C|$

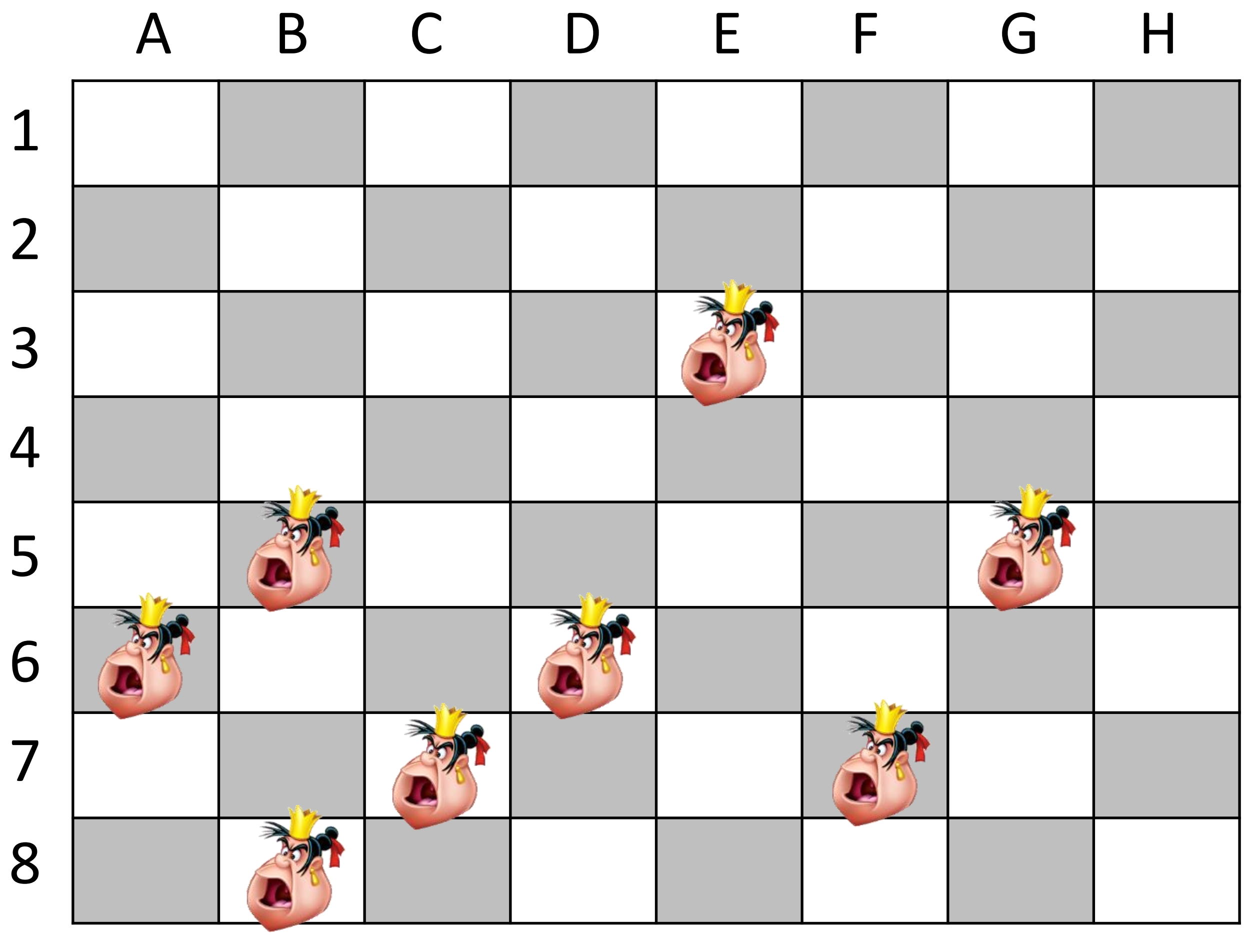


Continue…

Local Search Methods

Example: 8-Queens

Use a **hill-climbing** with the evaluation function “number of queens which are threatened by another queen” in the 8-Queens problem.



Example: 8-Queens

- What is the current score for the evaluation function?

In its current state, every queen is challenged by another, so the evaluation function scores 8

- Write down 3 of the possible moves from this state to the goal one (queens can move anywhere)

We want to move one queen to any other place on the board where it is not threatened by any other queen, as this will reduce the number of threatened queens by at least one. If we can do so in such a way that it frees up another queen, then this will result as a bonus. Thus,

B8 –> H8 : score after move is 5, because of the queens in B5, C7 and D6
G5 –> G4 : score after move is 6, because of the queens in G4 and E3
C7 –> C8 : score after move is 7, bacause of the queen in F7

Example: 8-Queens

- Give an example of an illegal move (in the hill-climbing search)

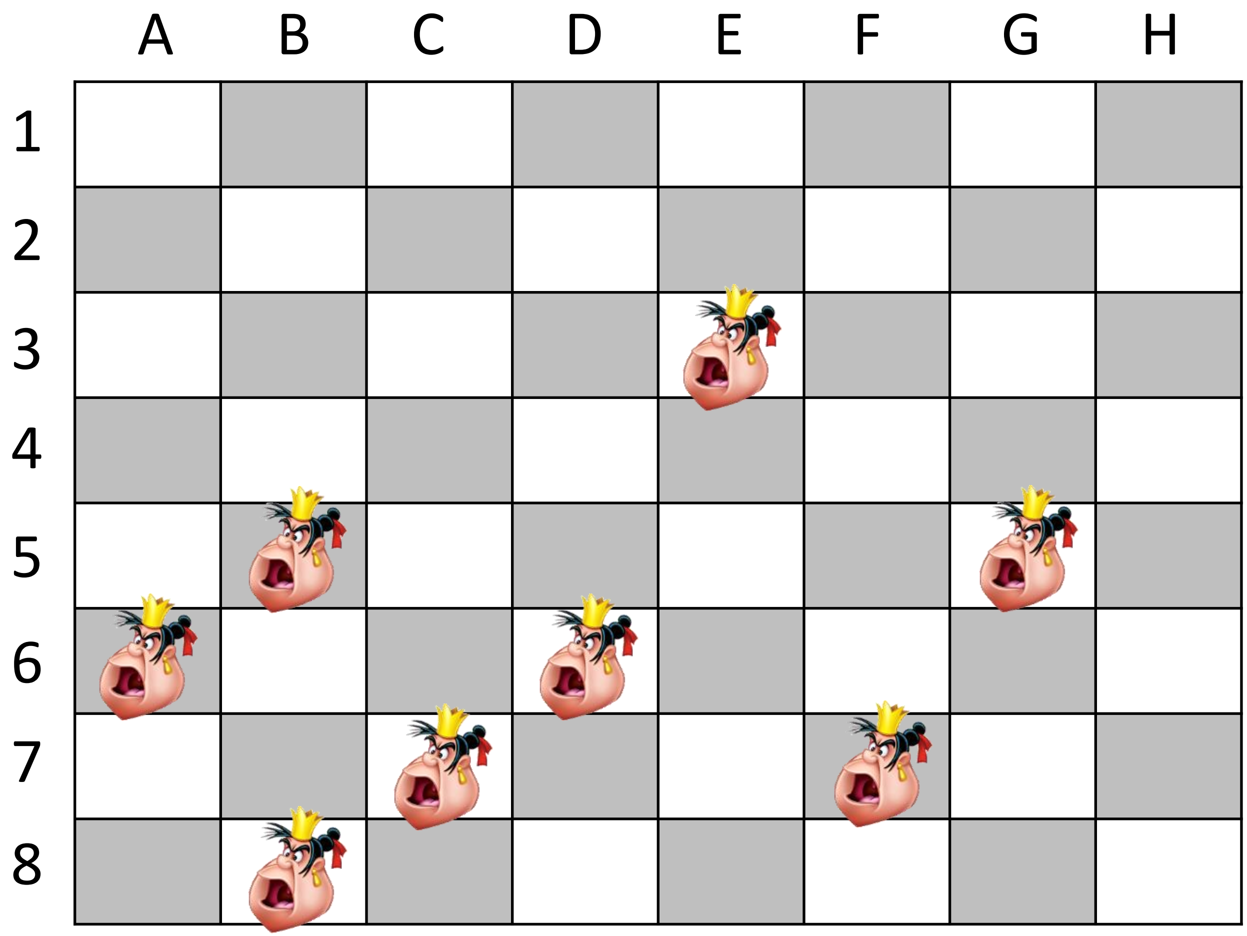
Hill-climbing searches require the evaluation function to strictly decrease after every move. For example, move B5 –> B3 is illegal since it would not decrease the number of threatened queens

- What do you do if there are no legal moves?

"Give up" when it happens. Start again by randomly putting your 8 queens on the board. This is called ***random-restart*** *and, no backtracking is needed. Hence, it is very good on memory efficiency, as only the current board state has to be kept in memory*

Local search: 8-Queens Hill Climbing

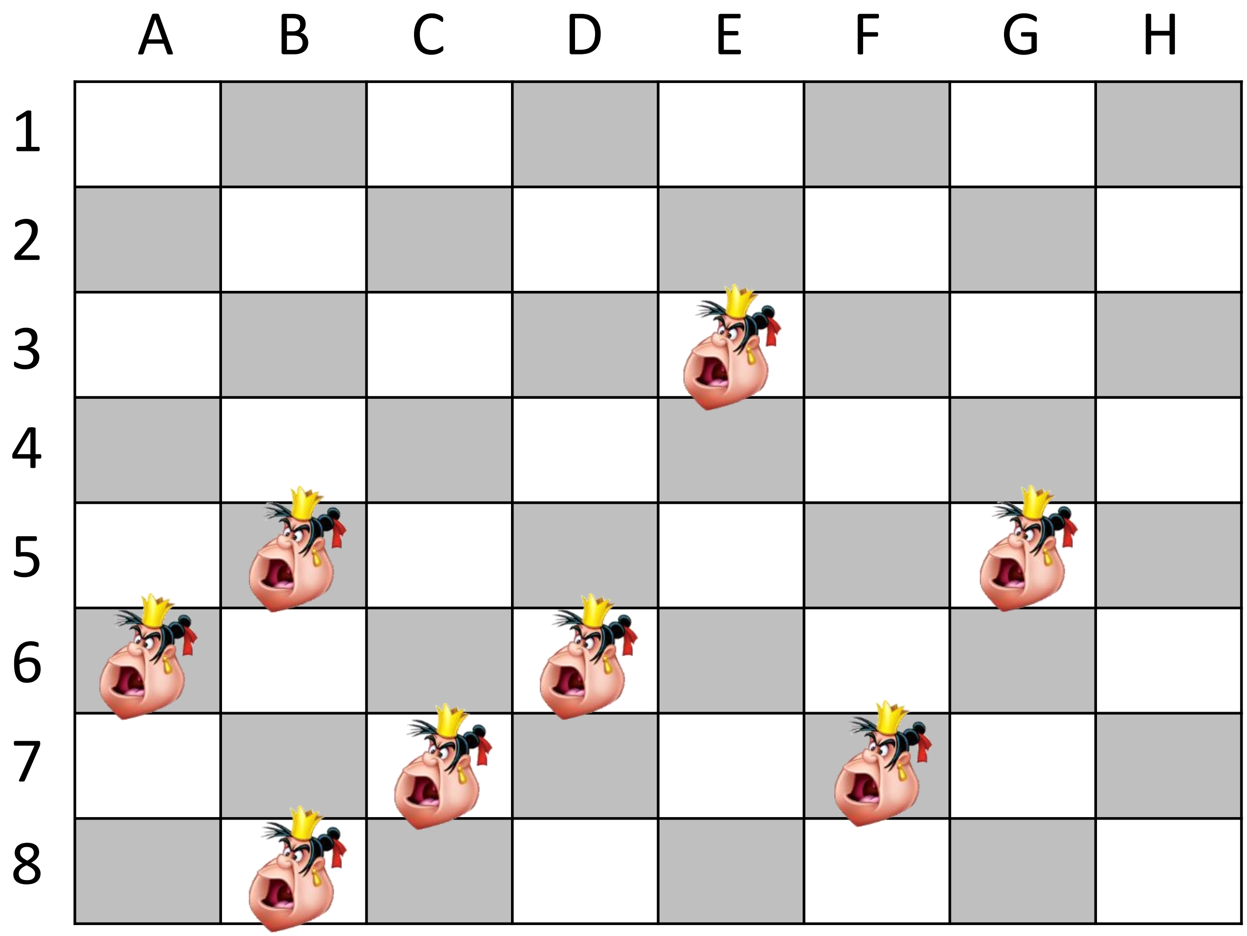
while *evaluation*(next_state) > *evaluation*(current_state) {



}

Local search: 8-Queens Hill Climbing

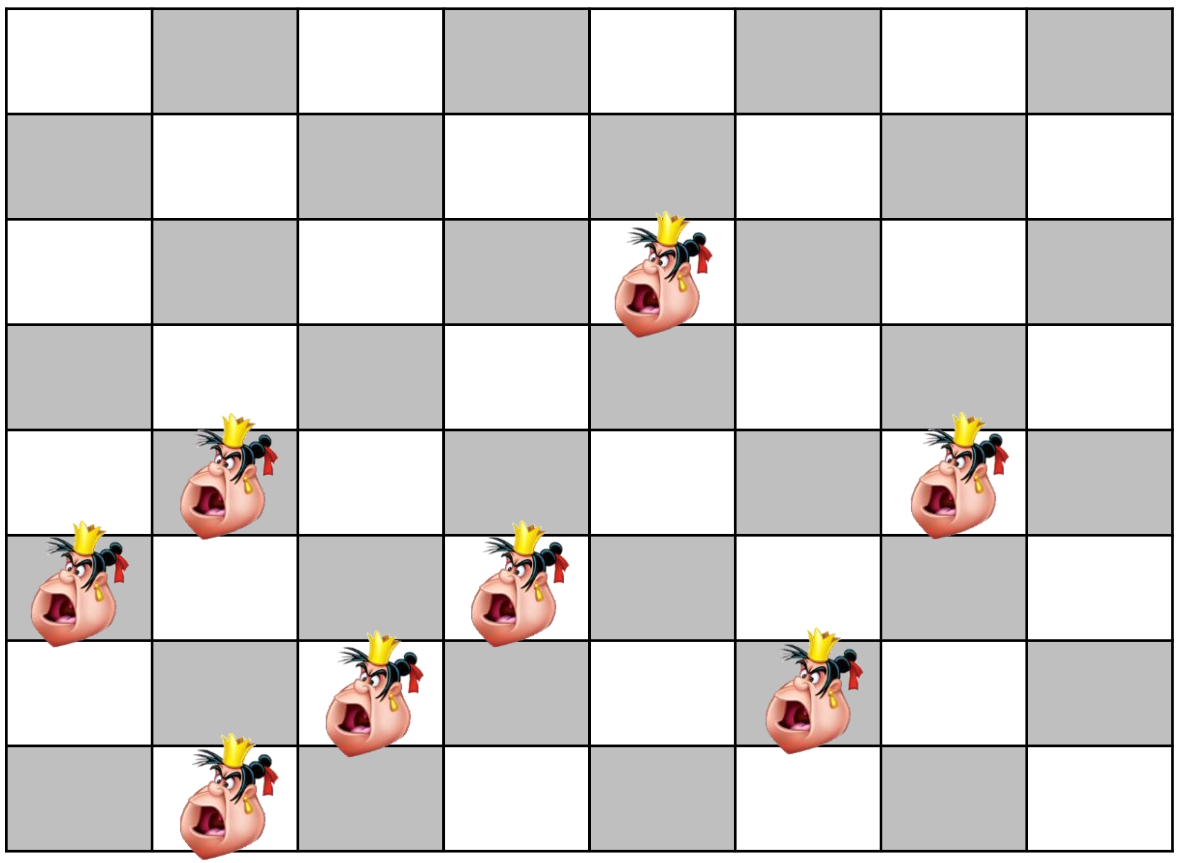
Best move after **first iteration** is: ***B8 –> H8***



Local search: 8-Queens Local Beam

At each iteration the algorithm keeps the first **K** best successor states.

K=2

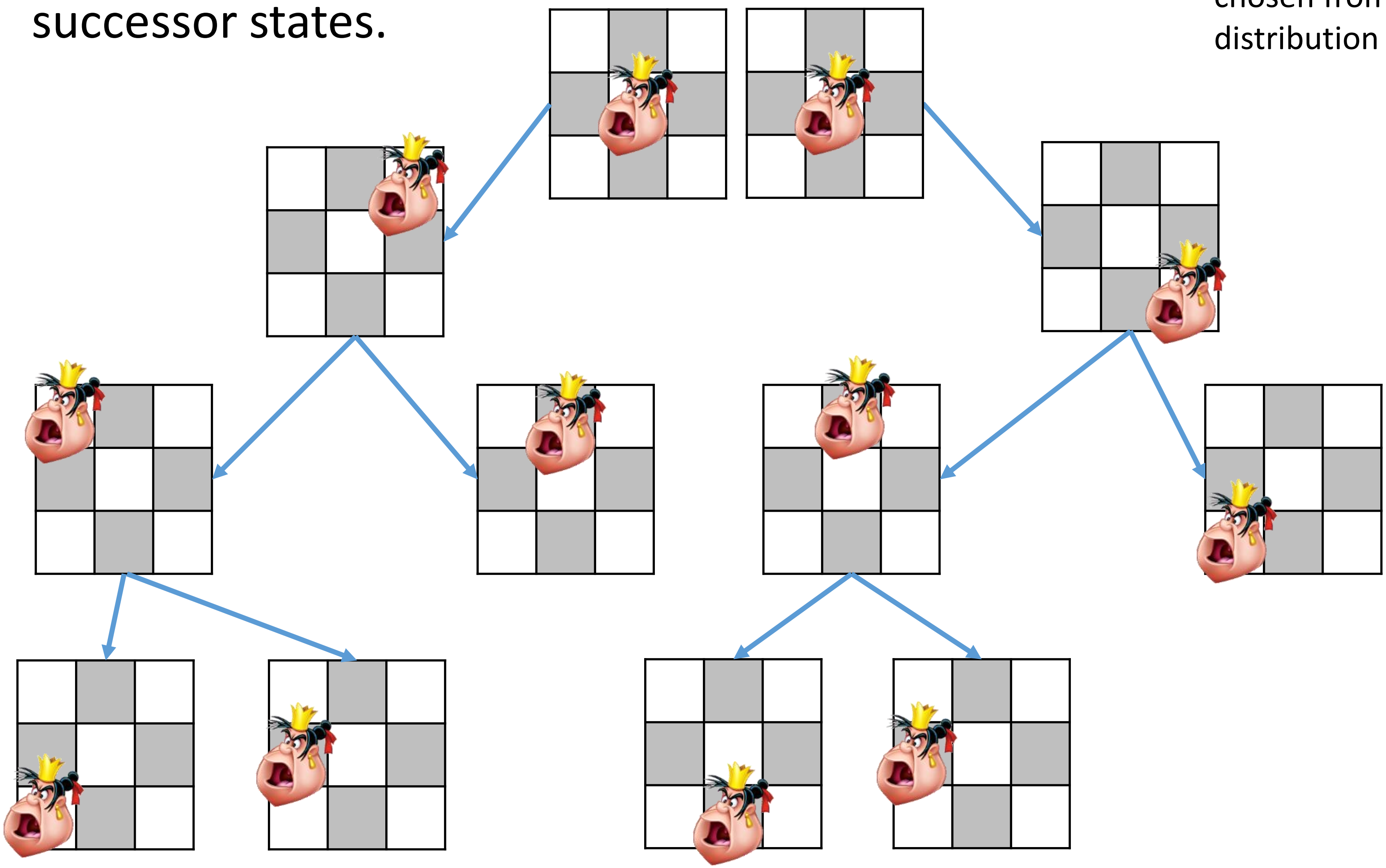


Local search: 8-Queens Local Beam

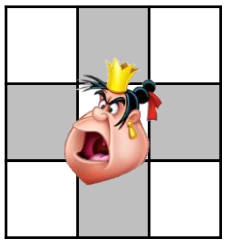
K=2

At each iteration the algorithm keeps the first **K** best successor states.

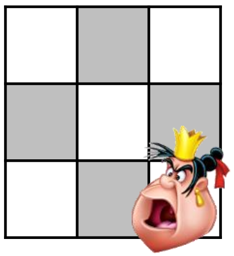
Where the initial states are randomly chosen from a state distribution



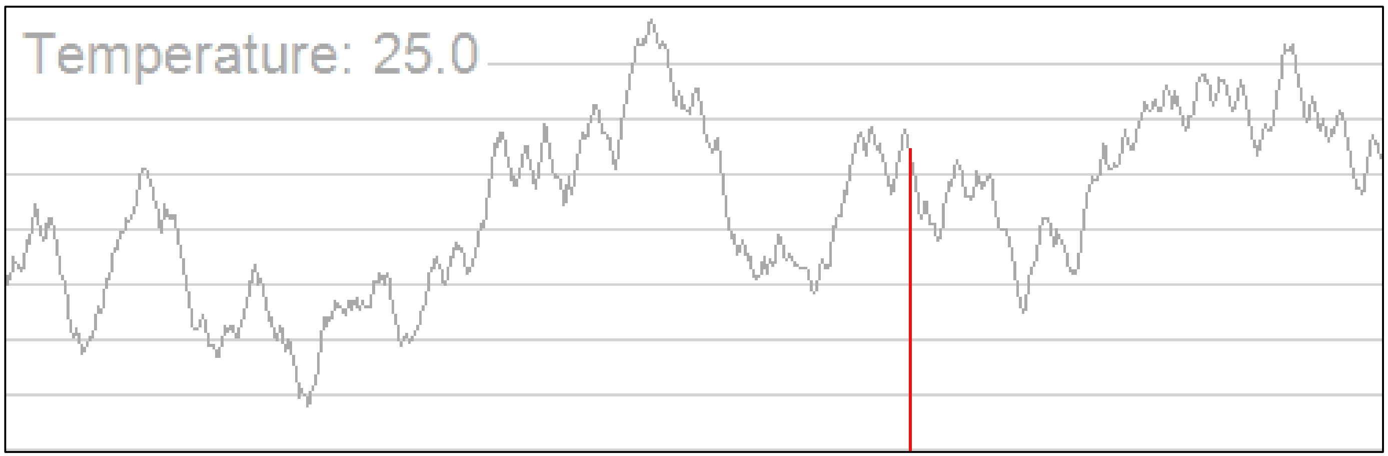
Local search: 8-Queens Simulated Annealing



If *value*(cost(*I*), cost(*N*), *T*) < *random*()

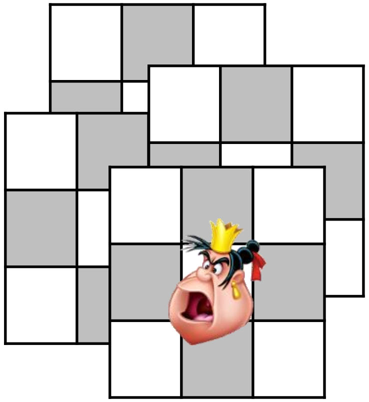


random initial state *I*	$e^{\frac{\mathrm{cost(I)} - \mathrm{cost(N)}}{T}}$	random next state *N*
cost(*I*)		cost(*N*)



*from wikipedia

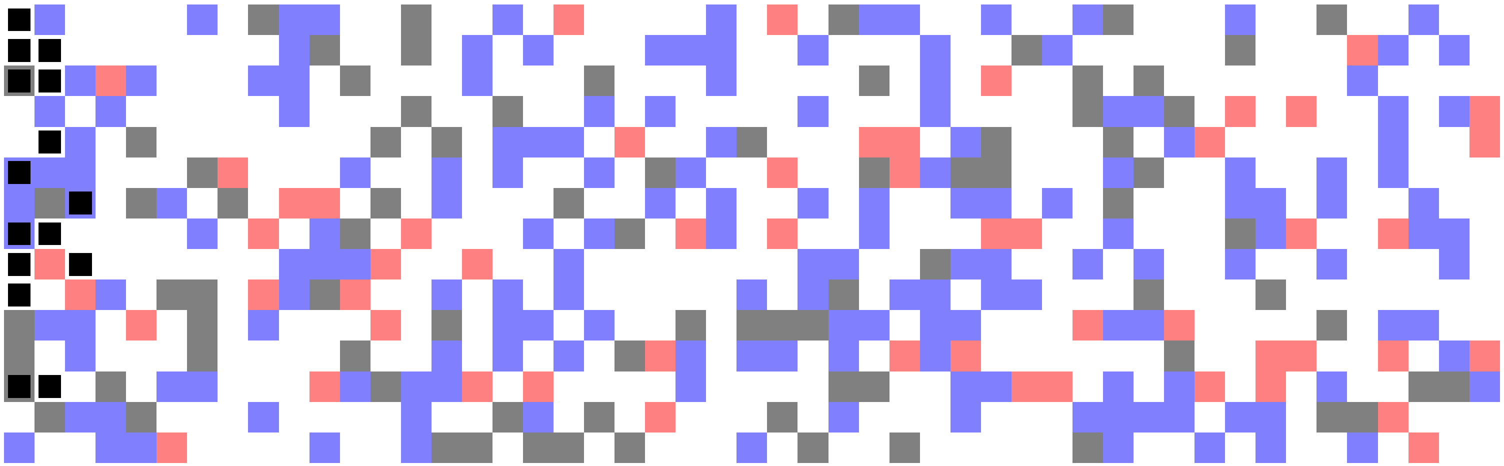
Local search: 8-Queens Genetic Algorithm



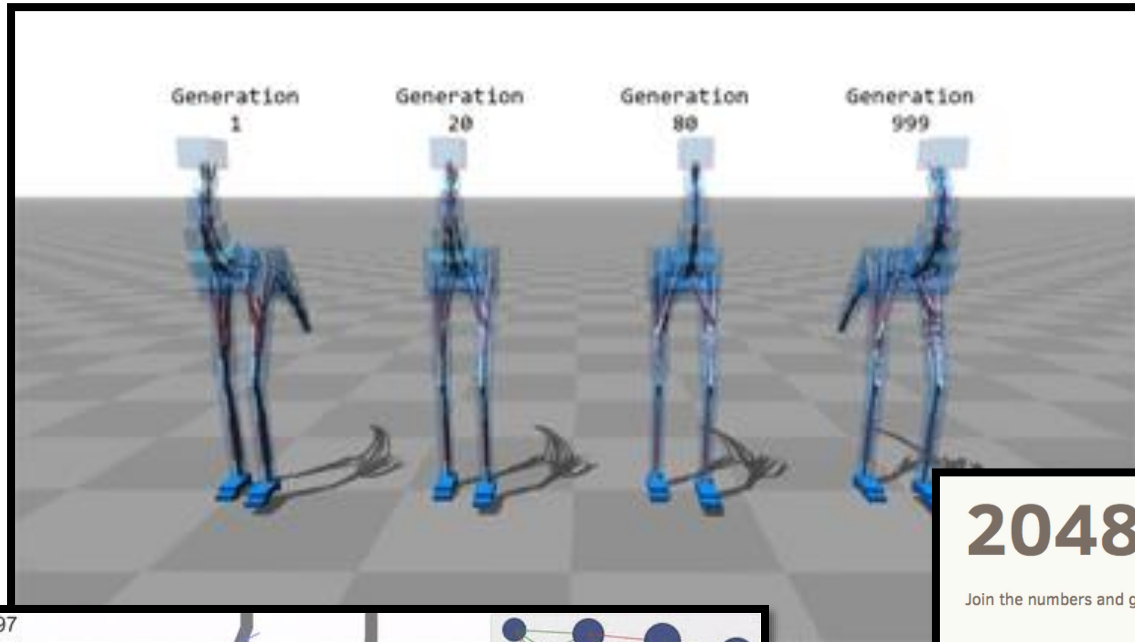
Parents selection wrt fitness function

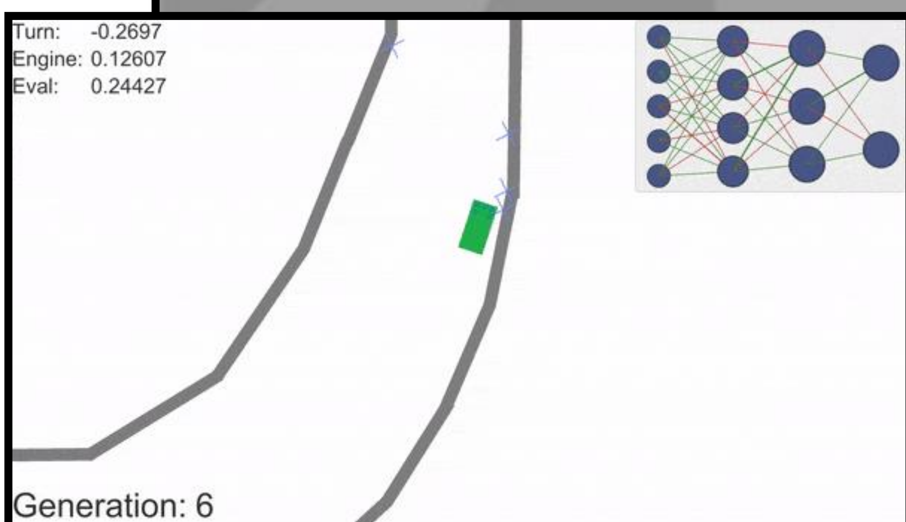
Reproduction by mutations and crossovers

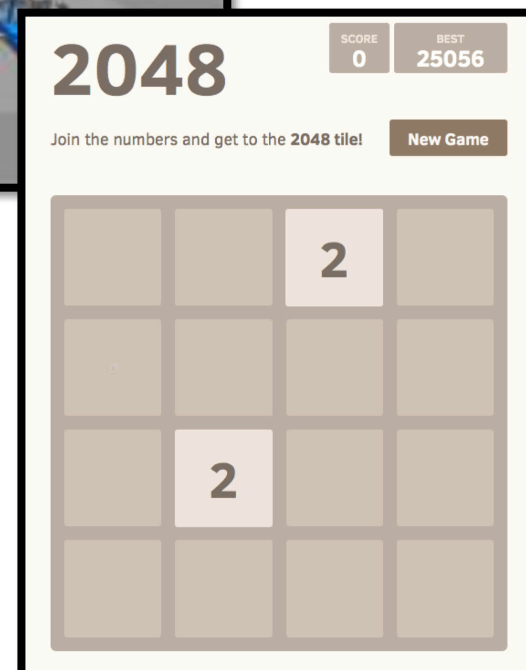
...repeat for *I* iterations



Local Search Algorithms







Local search: Sudoku

Goal: a complete board with numbers from 1 to 4, without conflicts

Rules: each of the 4 blocks has to contain all the numbers 1-4 within its squares. Each number can only appear once in a row, column or box.

Action: *change one digit at time*

	2	4	
1			3
4			2
	1	3	

Local search: Sudoku Hill Climbing

$step_0$

s_0:

4	2	4	3
1	1	1	3
4	4	2	2
2	1	3	3

$cost(s_0) = 11$

$$s = \left\langle \begin{matrix} \langle b_0, b_1, b_2, b_3 \rangle_0, \langle b_0, b_1, b_2, b_3 \rangle_1, \\ \langle b_0, b_1, b_2, b_3 \rangle_2, \langle b_0, b_1, b_2, b_3 \rangle_3 \end{matrix} \right\rangle$$

$cost(s)$
$= \#rows\ conflicts + \#cols\ conflicts$
$+ \#blocks\ concflicts$

$step_1$

s_1:

4	2	4	3
1	1	1	3
4	4	2	2
2	1	3	4

$cost(s_1) = 10$

, s_2:

4	2	4	3
1	1	1	3
4	4	2	2
4	1	3	3

$cost(s_2) = 11$

, s_3:

4	2	4	3
1	1	1	3
4	4	4	2
2	1	3	3

$cost(s_3) = 12$

, s_4:

1	2	4	3
1	1	1	3
4	4	2	2
2	1	3	3

$cost(s_4) = 10$

, s_5:

4	2	4	3
1	1	1	3
4	4	3	2
2	1	3	3

$cost(s_5) = 12$

....

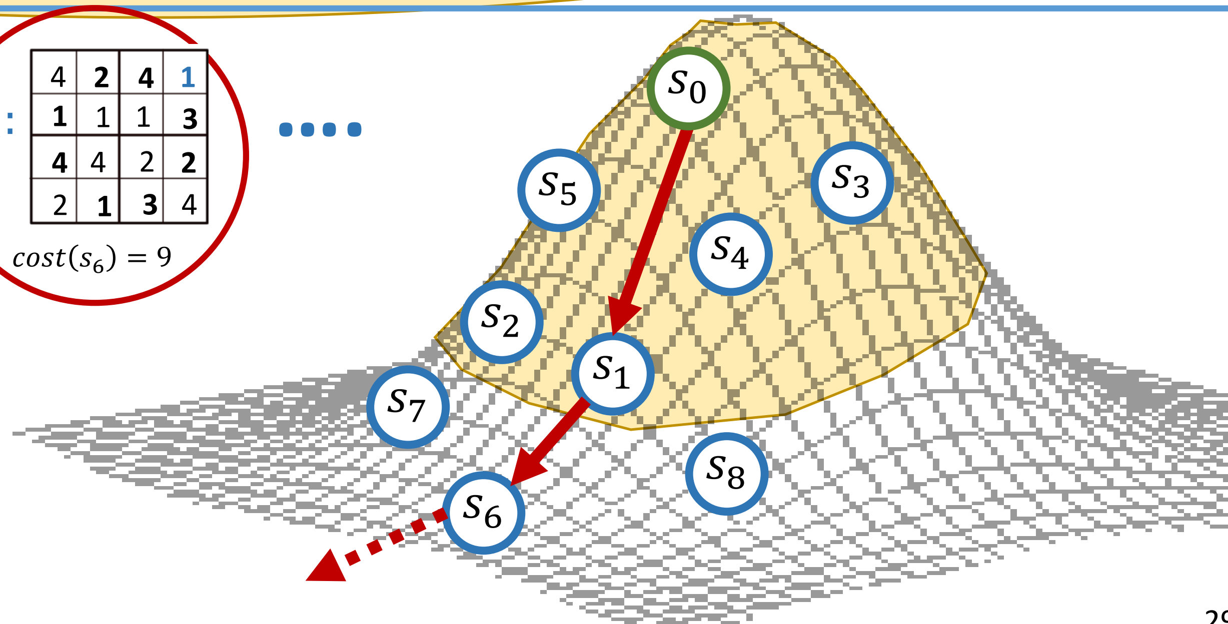
$step_2$

.... s_6:

4	2	4	1
1	1	1	3
4	4	2	2
2	1	3	4

$cost(s_6) = 9$

....



At each iteration you can change the value of the cells to [1,2,3,4]

Local search: Sudoku Hill Climbing

$step_2$

.... s_6 ....

4	**2**	**4**	1
1	1	1	**3**
4	4	2	**2**
2	**1**	**3**	4

$cost(s_6) = 9$

$step_3$

s_{10}:

4	**2**	**4**	1
1	1	2	**3**
4	4	2	**2**
2	**1**	**3**	4

$cost(s_{10}) = 9$

, s_{11}:

4	**2**	**4**	1
1	1	3	**3**
4	4	2	**2**
2	**1**	**3**	4

$cost(s_{11}) = 10$

.... , s_{14}:

4	**2**	**4**	1
1	1	1	**3**
4	3	2	**2**
2	**1**	**3**	4

$cost(s_{14}) = 8$

....

$step_4$

s_{18}:

2	**2**	**4**	1
1	1	1	**3**
4	3	2	**2**
2	**1**	**3**	4

$cost(s_{18}) = 8$

$cost(s_{22}) = 13$

, s_{19}:

3	**2**	**4**	1
1	1	1	**3**
4	3	2	**2**
2	**1**	**3**	4

$cost(s_{19}) = 6$

$cost(s_{22}) = 8$

.... , s_{22}

4	**2**	**4**	1
1	3	1	**3**
4	3	2	**2**
2	**1**	**3**	4

$cost(s_{22}) = 8$

$cost(s_{22}) = 8$

....

$$cost(s) = cost(s) + \sum_{d=1}^{4} |4 - \#d|$$

Where #d is the number of times a particular digit occurs

Local search: Sudoku Local Beam

$step_0$

4	2	4	3
1	1	1	3
4	4	2	2
2	1	3	3

$s_{0:K1}$: $cost = 11$

2	2	4	3
1	1	1	3
4	4	1	2
2	1	3	3

$s_{0:K2}$: $cost = 11$

$$s = \left\langle \begin{matrix} \langle b_0, b_1, b_2, b_3\rangle_0, \langle b_0, b_1, b_2, b_3\rangle_1, \\ \langle b_0, b_1, b_2, b_3\rangle_2, \langle b_0, b_1, b_2, b_3\rangle_3 \end{matrix} \right\rangle$$

$cost(s)$

$= \#rows\ conflicts + \#cols\ conflicts$

$+ \#blocks\ concflicts$

$step_1$

s_1:

4	2	4	3
1	1	1	3
4	4	2	2
2	1	3	4

$cost(s_1) = 10$

s_2:

4	2	4	3
1	1	1	3
4	4	2	2
4	1	3	3

$cost(s_2) = 11$

s_3:

4	2	4	3
1	1	1	3
4	4	4	2
2	1	3	3

$cost(s_3) = 12$

s_4:

1	2	4	3
1	1	1	3
4	4	2	2
2	1	3	3

$cost(s_4) = 10$

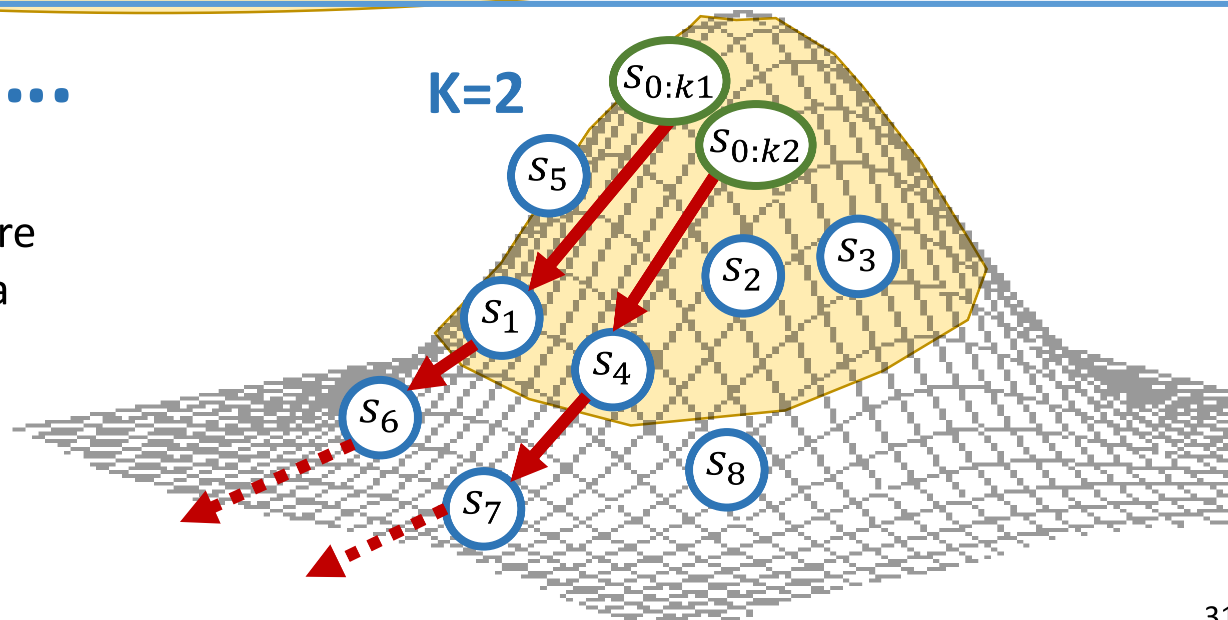
s_5:

4	2	4	3
1	1	1	3
4	4	3	2
2	1	3	3

$cost(s_5) = 12$

....

....



Where $s_{0:k1}$ and $s_{0:k2}$ are randomly chosen from a state distribution

Examples

Learning motions with **policy gradient** and **genetic algorithm**