Artificial Intelligence
Informed search

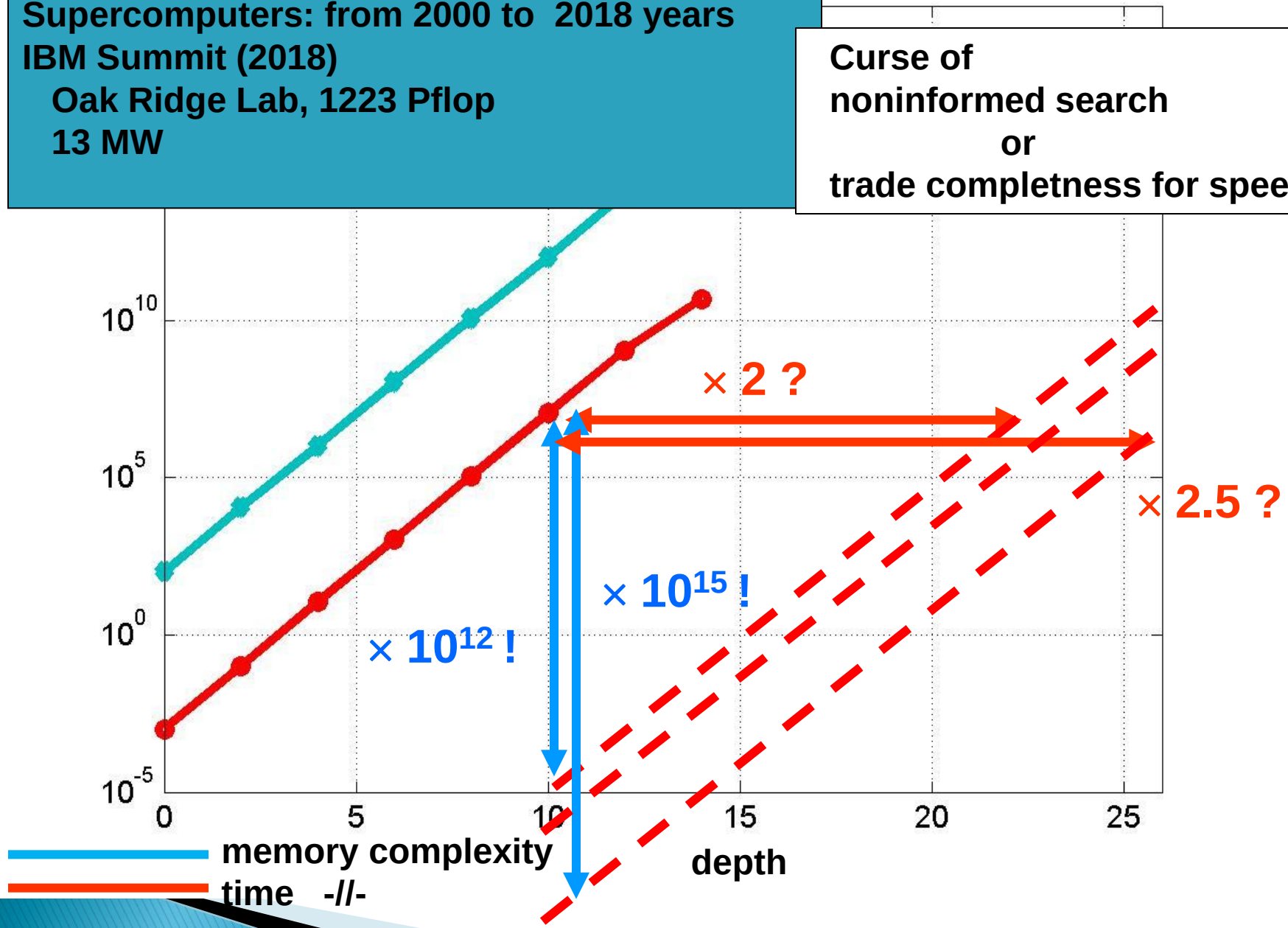
Peter Antal
antal@mit.bme.hu
Tadeusz Dobrowiecki
tade@mit.bme.hu

Outline

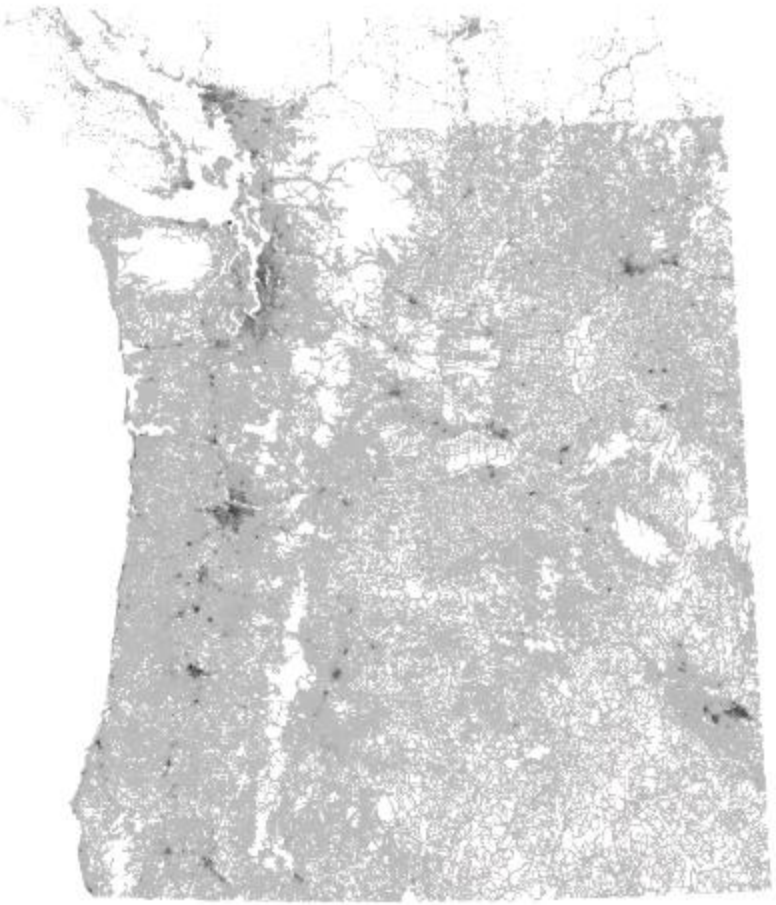
- Informed = use problem-specific knowledge
- Which search strategies?
  - Best-first search and its variants
- Heuristic functions?
  - How to invent them
- Local search and optimization
  - Hill climbing, local beam search, genetic algorithms,…

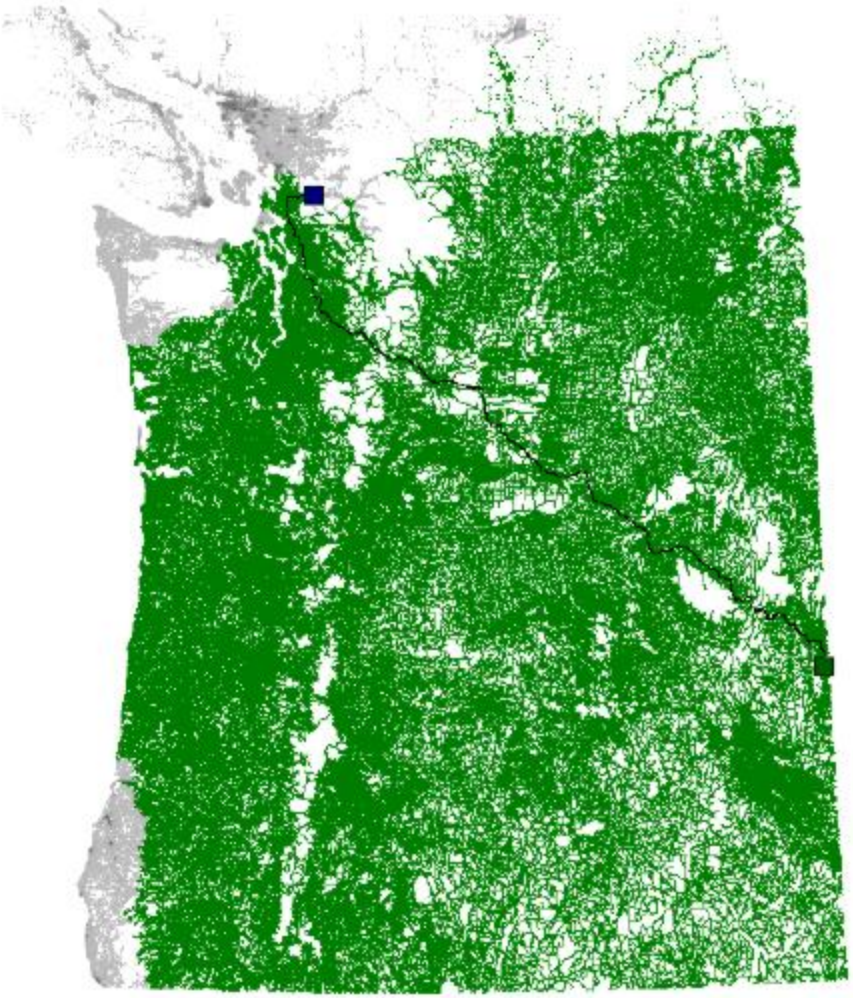
Supercomputers: from 2000 to 2018 years
IBM Summit (2018)
Oak Ridge Lab, 1223 Pflop
13 MW

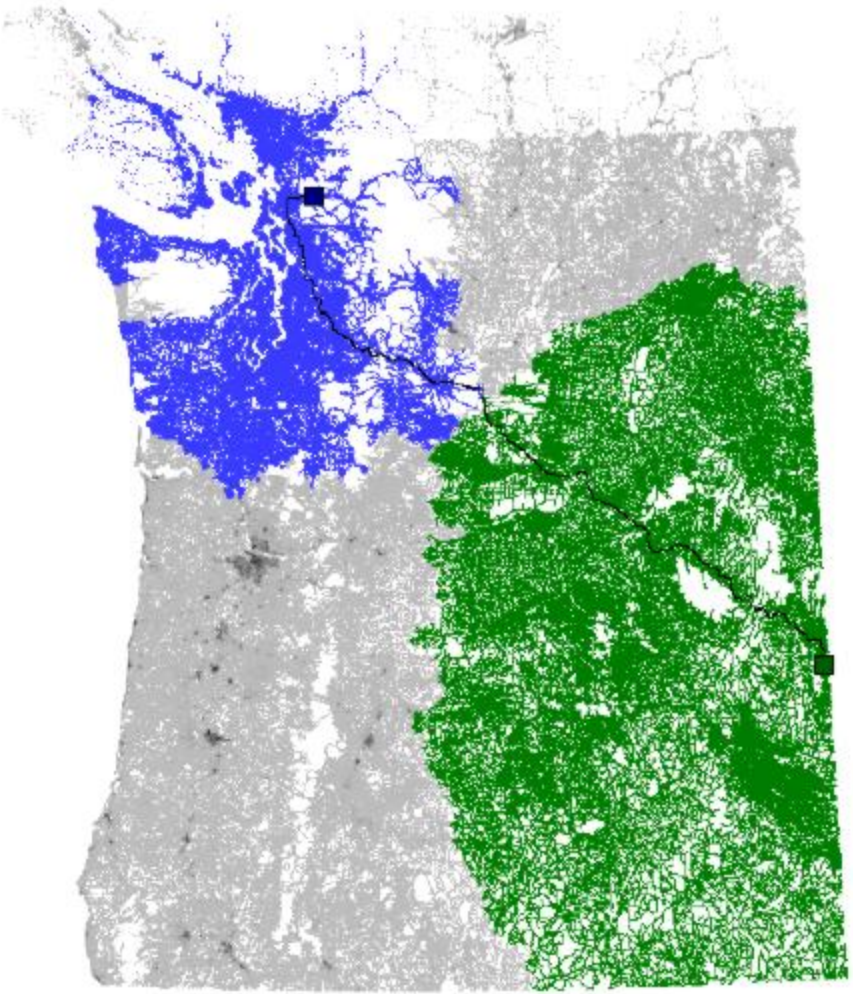
Curse of
noninformed search
or
trade completness for speed

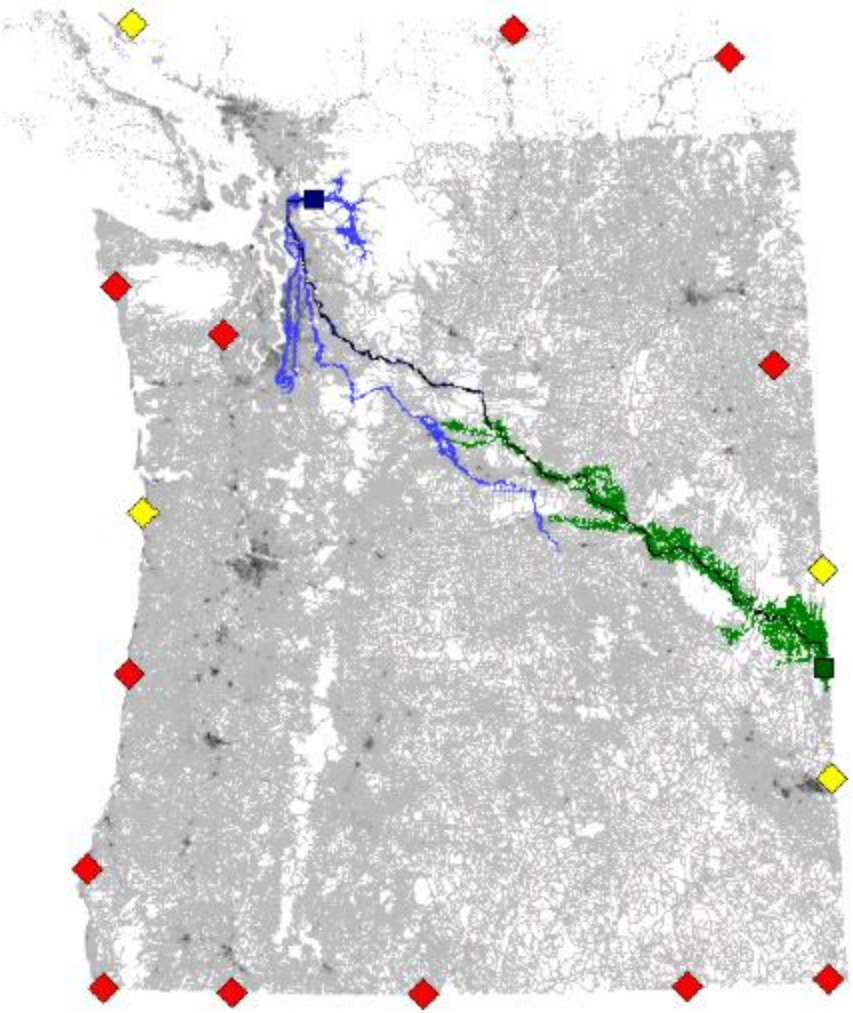


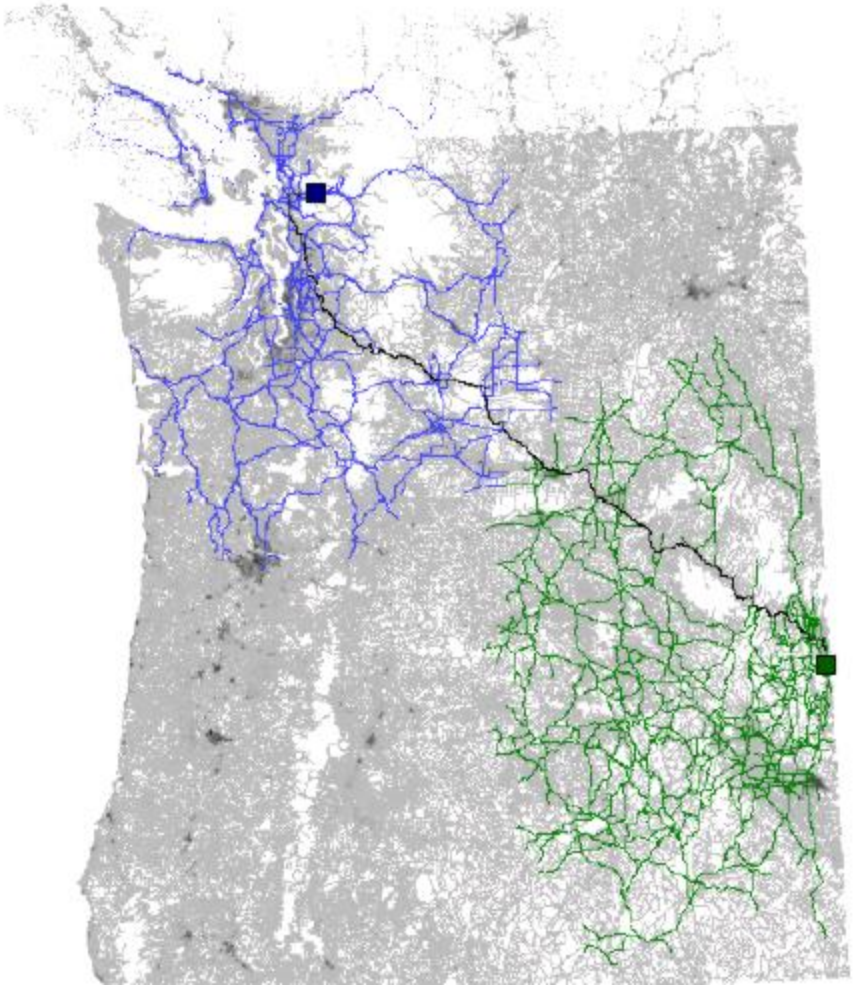
1.6M vertices
3.8M edges

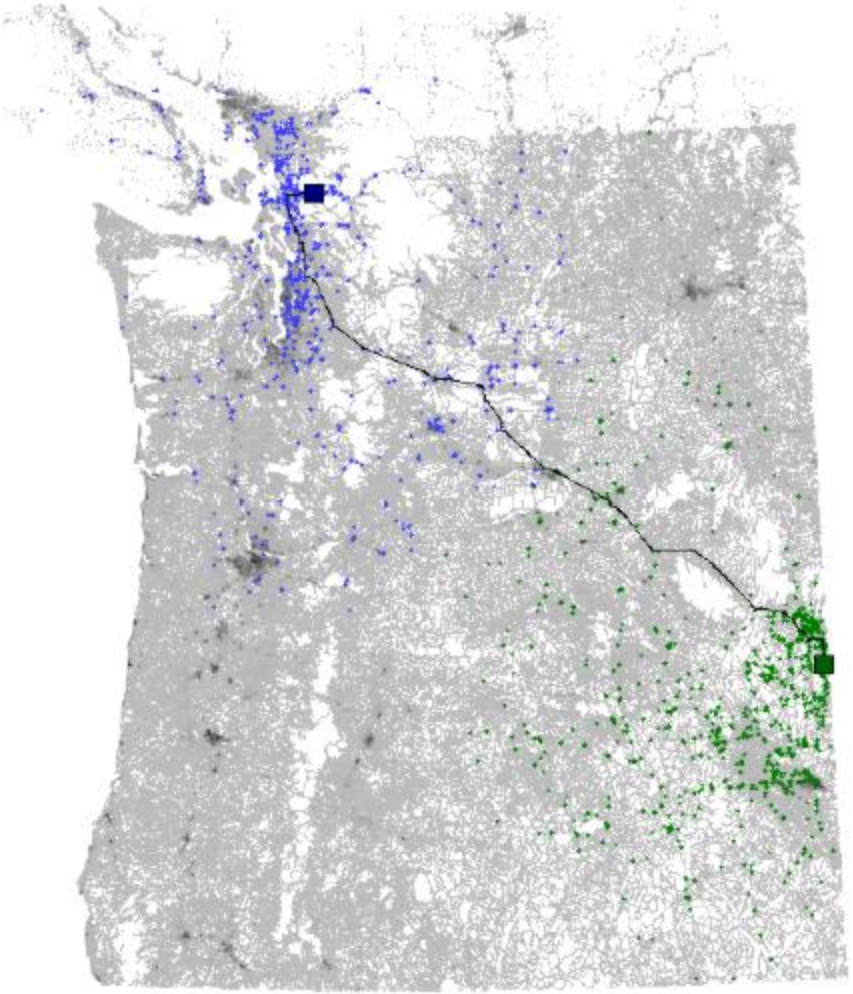












(1) Penalty for backtracking. (A*, etc.)

(2) Search from both directions. (bidirectional)

(3) Smart usage of depth-first search. (iteratíve)

(4) Pre-process the graph. (ALT landmarks)

(5) Start with small memory, in principle. (EMA*)

(6) Fully give up backtracking. (hill-climbing)

(7) Use lower quality directions also. (beam search)

(8) Permit bad directions also. (simulated annealing)

(9) Prohibit probably bad directions. (tabu search)

(10) Continuously improve based on partial results.
(learning A*, anytime A*, etc.)

(11) Randomize and repeat.

(12) Stay and search in the original continuous physical space. ...

Penalty for backtracking!

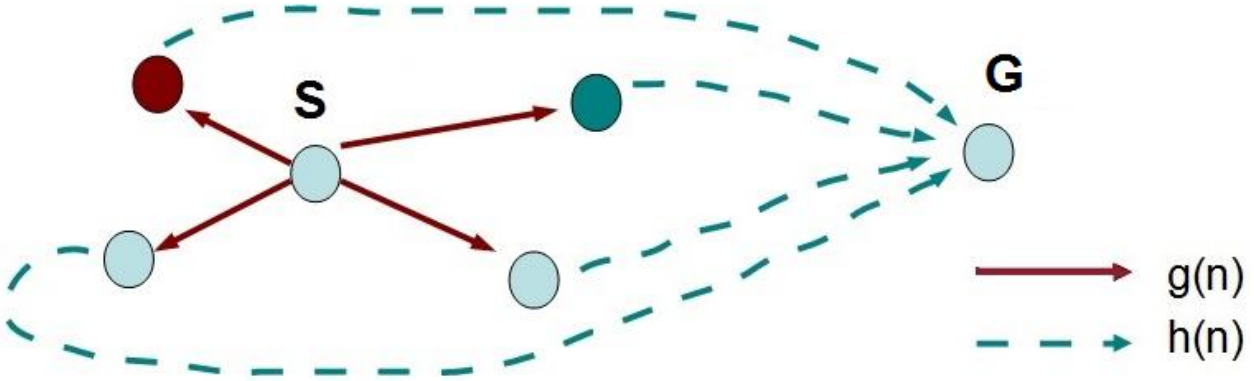
= awarding going forward, **toward the goal**.

What is needed: - In what „direction” to expect the goal?
- How „far” to expect the goal in this direction?.

This information is called **heuristics**, **heuristic function h(n)**:

- Must be computable in every state in the searc space
- Estimates the expected cost of going in a given direction
- If exact, makes the search superficial (if very uncertain, does not help at all)
- At goal is zero: h(goal) = 0

Search using heuristics is a **heuristic** (**informed**) **search**.

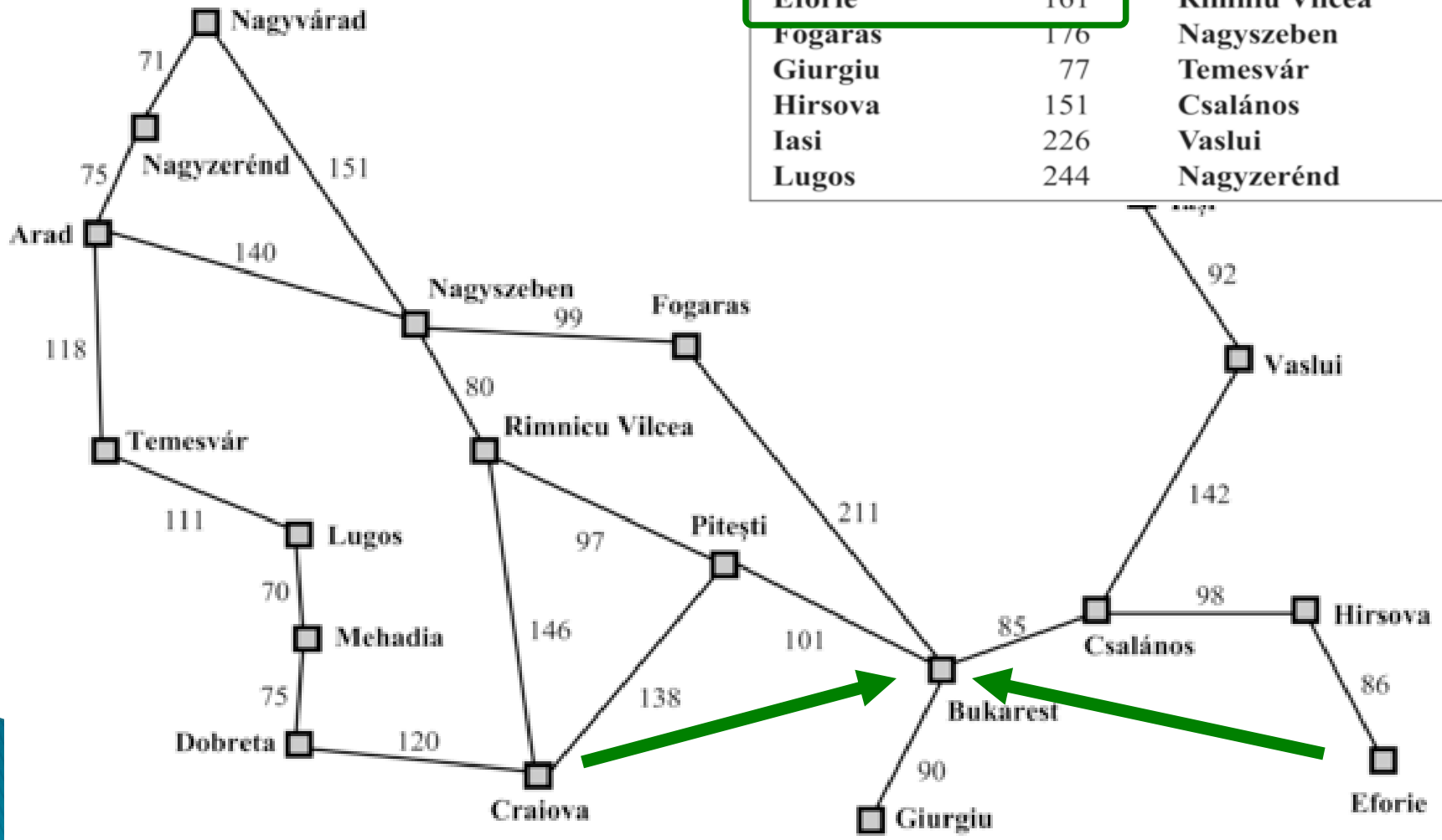


Let the heuristics – straightline distance (h_{SLD})

Fullfills conditions?

What about its error?

Arad	366	Mehadia	241
Bukarest	0	Neamt	234
Craiova	160	Nagyvárad	380
Dobreta	242	Piteşti	100
Eforie	161	Rimniu Vilcea	193
Fogaras	176	Nagyszeben	253
Giurgiu	77	Temesvár	329
Hirsova	151	Csalános	80
Iasi	226	Vaslui	199
Lugos	244	Nagyzerénd	374

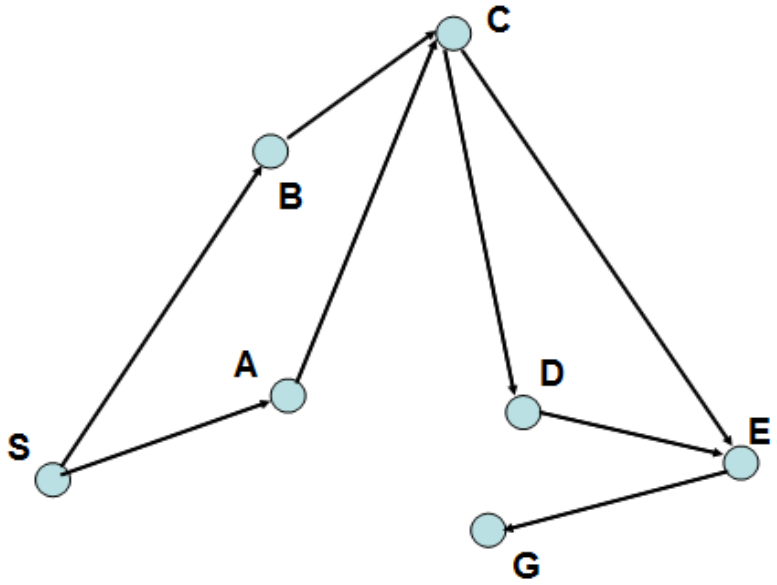


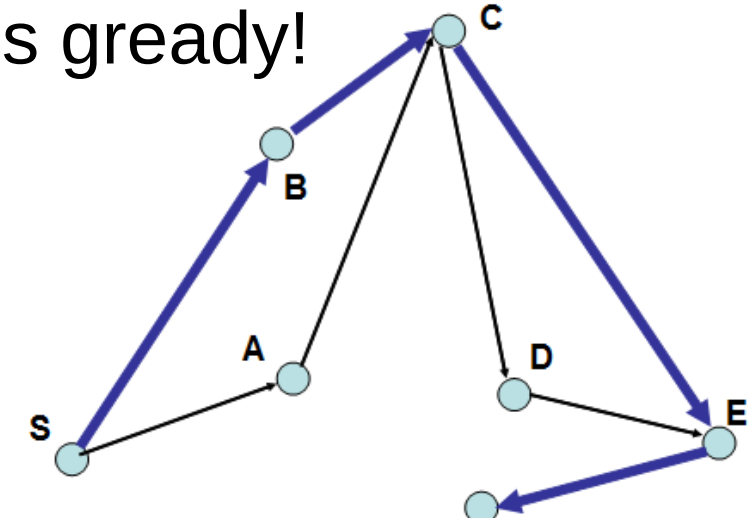
Best-first search

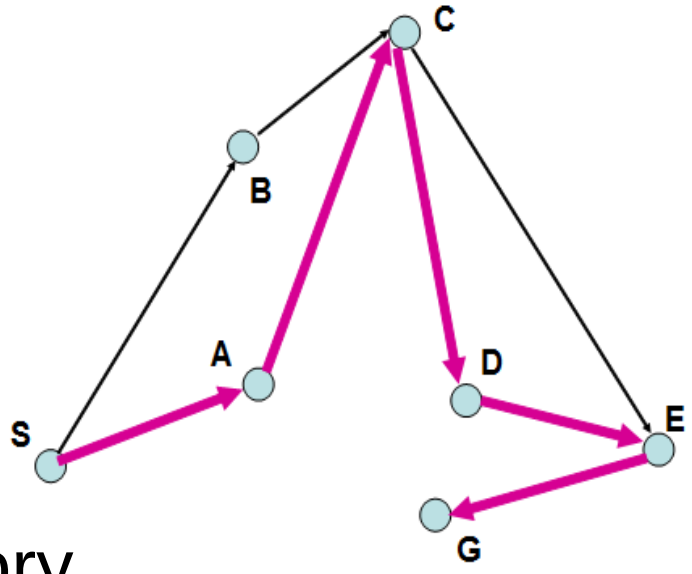
- General approach of informed search:
  - Best-first search: node is selected for expansion based on an *evaluation function* $f(n)$ in TREE-SEARCH().
- Idea: evaluation function measures distance to the goal.
  - Choose node which *appears* best
- Implementation:
  - *fringe* is queue sorted in decreasing order of desirability.
  - Special cases: greedy search, A* search

(Ro-MohoK)

Best-First with only h(n) is gready!







Complexity: time = memory

If exact h(n): linear time and memory, but if not?

A* search

- Best-known form of best-first search.
- Idea: avoid expanding paths that are already expensive.
- Evaluation function $f(n)=g(n) + h(n)$
  - $g(n)$ the cost (so far) to reach the node.
  - i.e. summa of action costs along the path
  - $h(n)$ estimated cost to get from the node to the closest goal.
  - $f(n)$ estimated total cost of path through n to goal.

A* search

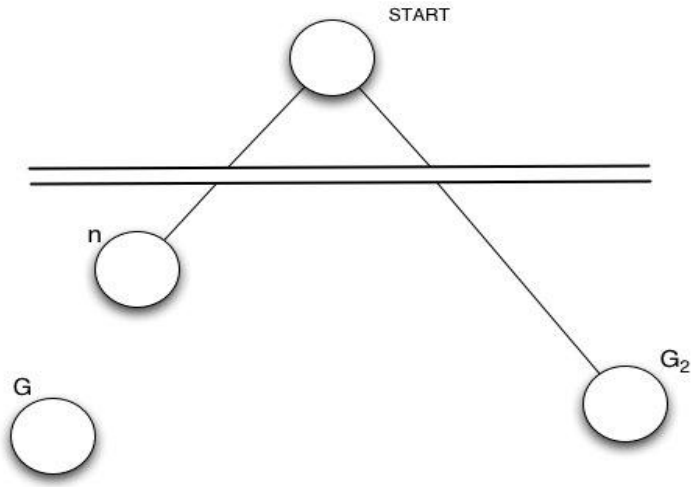
- A* search uses an admissible heuristic
  - A heuristic is *admissible* if it *never overestimates* the cost to reach the goal (~optimistic).

    Formally:
    1. $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from n
    2. $h(n) >= 0$ so $h(G)=0$ for any goal G.

    e.g. $h_{SLD}(n)$ never overestimates the actual road distance

    Theorem: If $h(n)$ is admissible, A* using `BEST-FIRST-SEARCH()` with selector function f(n) is optimal.

Optimality of A*(standard proof)



Suppose a suboptimal goal G_2 in the queue.
Let n be an unexpanded node on a shortest to optimal goal G.

$f(G_2) = g(G_2)$ since $h(G_2)=0$
$> g(G)$ since G_2 is suboptimal
$>= f(n)$ since h is admissible

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion (i.e. for checking, but note that G_2 can be inside the queue).

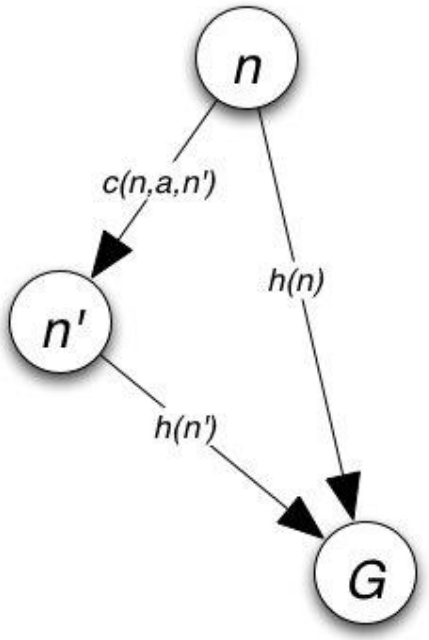
Consistency

- A heuristic is consistent if

$$h(n) \leq c(n,a,n') + h(n')$$

- If h is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n,a,n') + h(n') \\ &\geq g(n) + h(n) \\ &\geq f(n) \end{aligned}$$



i.e. f(n) is non-decreasing along any path.

Theorem: If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal

Optimality of A*(more usefull)

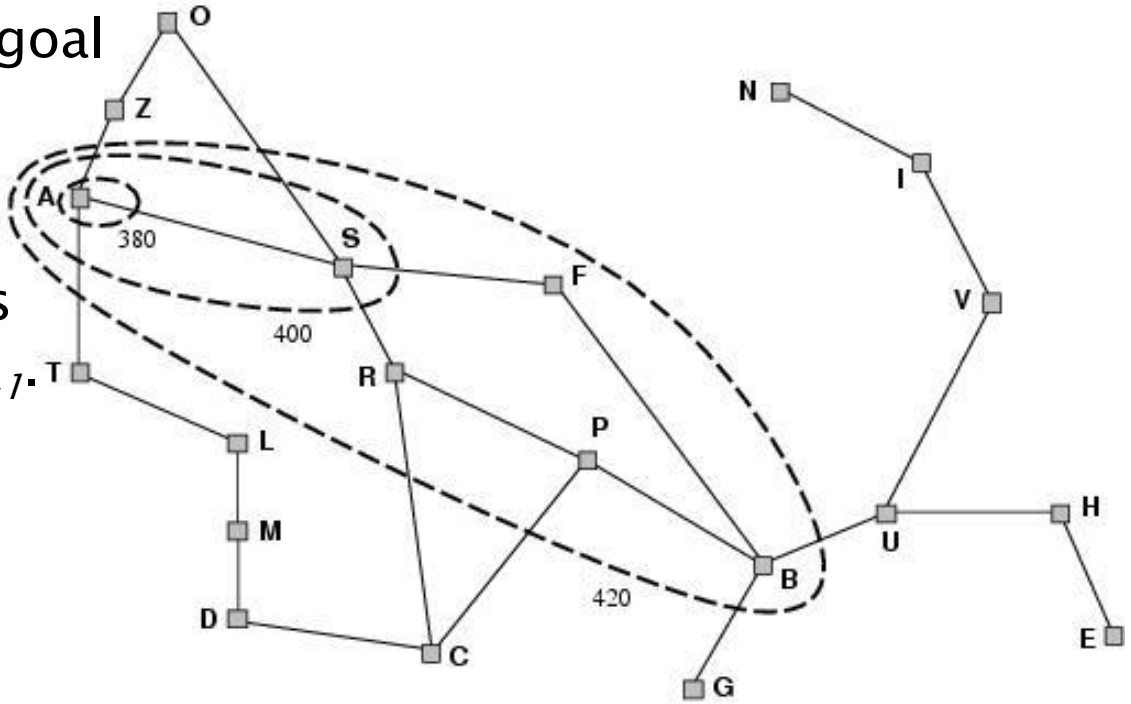
- A* expands nodes in order of increasing *f* value
- Contours can be drawn in state space
  - Uniform-cost search adds circles.
  - F-contours are gradually added:

1) nodes with $f(n)<C^*$

2) Some nodes on the goal

Contour ($f(n)=C^*$).

Contour *i* has all nodes with $f=f_i$, where $f_i < f_{i+1}$.



http://mialmanach.mit.bme.hu/demonstraciok/a_kereses_usa_terkepen_konturkovetessel

A* search, evaluation

- Completeness: YES
  - Since bands of increasing *f* are added
  - Unless there are infinitly many nodes with $f<f(G)$

Locally finite graphs – finite branching factor

– action cost $> \varepsilon > 0$.

A* search, evaluation

- Completeness: YES
- Time complexity:
  - Number of nodes expanded is still exponential in the length of the solution.

A* search, evaluation

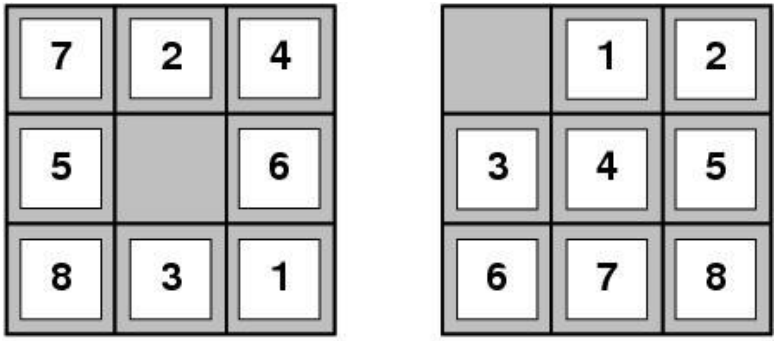
- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity:
  - It keeps all generated nodes in memory
  - Hence space is the major problem not time

A* search, evaluation

- Completeness: YES
- Time complexity: (exponential with path length)
- Space complexity:(all nodes are stored)
- Optimality: YES
  - Cannot expand f_{i+1} until f_i is finished.
  - A* expands all nodes with $f(n) < C^*$
  - A* expands some nodes with $f(n) = C^*$
  - A* expands no nodes with $f(n) > C^*$

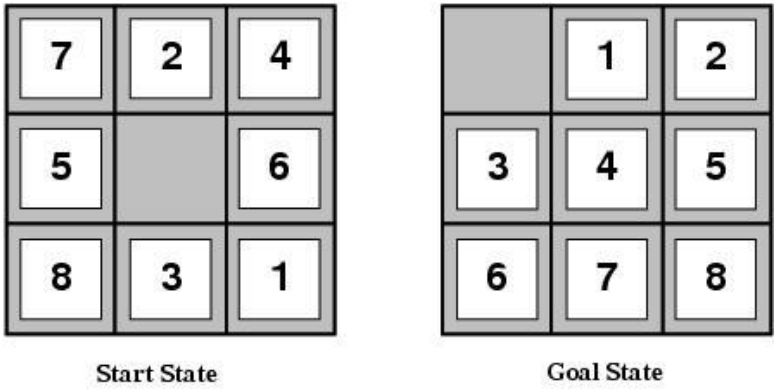
Also *optimally efficient* (not including ties)

Finding heuristic functions



- E.g for the 8-puzzle
  - Avg. solution cost is about 22 steps (branching factor +/- 3)
  - Exhaustive search to depth 22: 3.1 x 10^{10} states.
  - A good heuristic function can reduce the search process.

Finding heuristic functions



- E.g for the 8-puzzle knows two commonly used heuristics
- h_1 = the number of misplaced tiles
  - $h_1(s)=8$
- h_2 = the sum of the distances of the tiles from their goal positions (manhattan distance).
  - $h_2(s)=3+1+2+2+2+3+3+2=18$

Quality (error) of heuristic functions

- Effective branching factor b*
  - Is the branching factor that a uniform tree of depth *d* would have in order to contain *N+1* nodes.

$$N+1=1+b^*+(b^*)^2+\ldots+(b^*)^d$$

  - Measure is fairly constant for sufficiently hard problems.
    - Can thus provide a good guide to the heuristic's overall usefulness.
    - A good value of b* is 1.

Heuristic quality and dominance

- 1200 random problems with solution lengths from 2 to 24.

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

- If $h_2(n) >= h_1(n)$ for all n (both admissible) then h_2 *dominates* h_1 and is better for search

- h_1 = the number of misplaced tiles
- h_2 = the sum of the distances of the tiles from their goal positions (manhattan distance)

	Search Cost			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Inventing admissible heuristics

- Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem:
  - Relaxed 8-puzzle for h_1 : a tile can move anywhere
    As a result, $h_1(n)$ gives the shortest solution
  - Relaxed 8-puzzle for h_2 : a tile can move to any adjacent square.
    As a result, $h_2(n)$ gives the shortest solution.

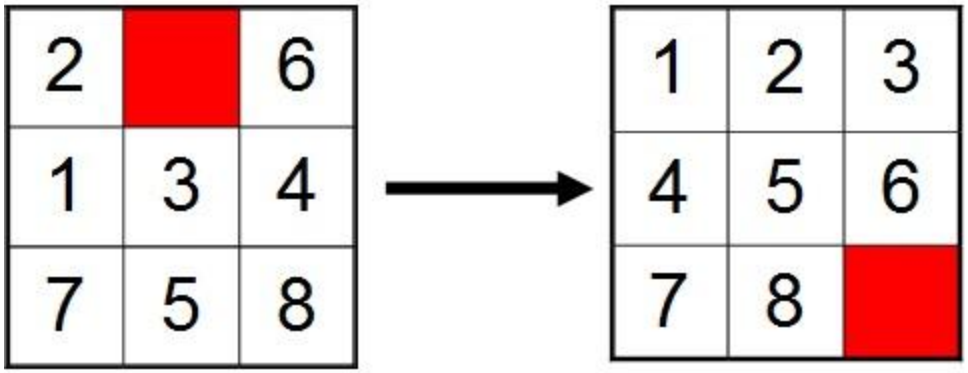
The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem.

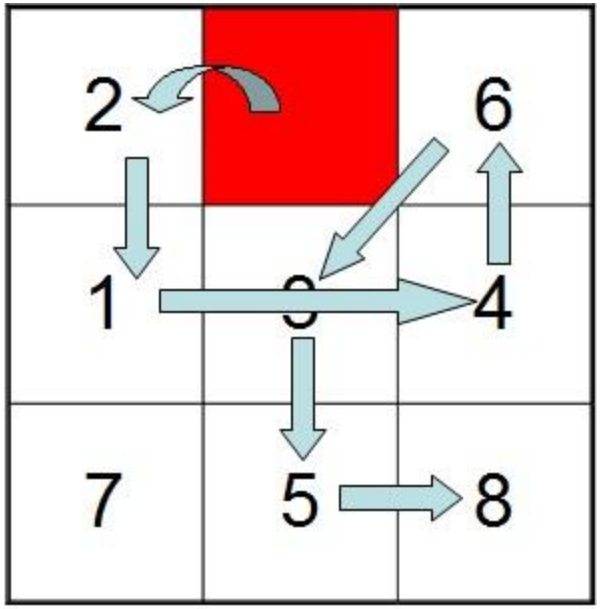
The optimal solution of a relaxed problem is an admissible heuristics for the real problem.

Inventing admissible heuristics

Gaschnig's Heuristics

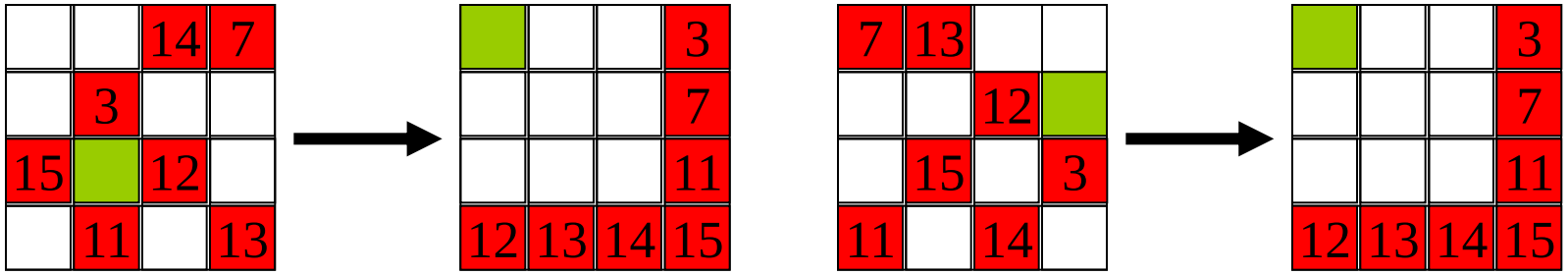
(b) A tile can be moved from A to B,
if B is empty.
admissible, good estimate





Inventing admissible heuristics

- Admissible heuristics can also be derived from the solution cost of a subproblem of a given problem.
- This cost is a lower bound on the cost of the real problem.
- Pattern databases store the exact solution for every possible subproblem instance.
  - The complete heuristic is constructed using the patterns in the DB



cost(subproblem1) + cost(subproblem2)
=< cost(subproblem1 + subproblem2)

Inventing admissible heuristics

- $h(n) = \max\{ h_1(n),\ldots,h_m(n) \}$

- Another way to find an admissible heuristic is through learning from experience:
  - Experience = solving lots of 8-puzzles
  - An inductive learning algorithm can be used to predict costs for other states that arise during search.

- Cost of computation

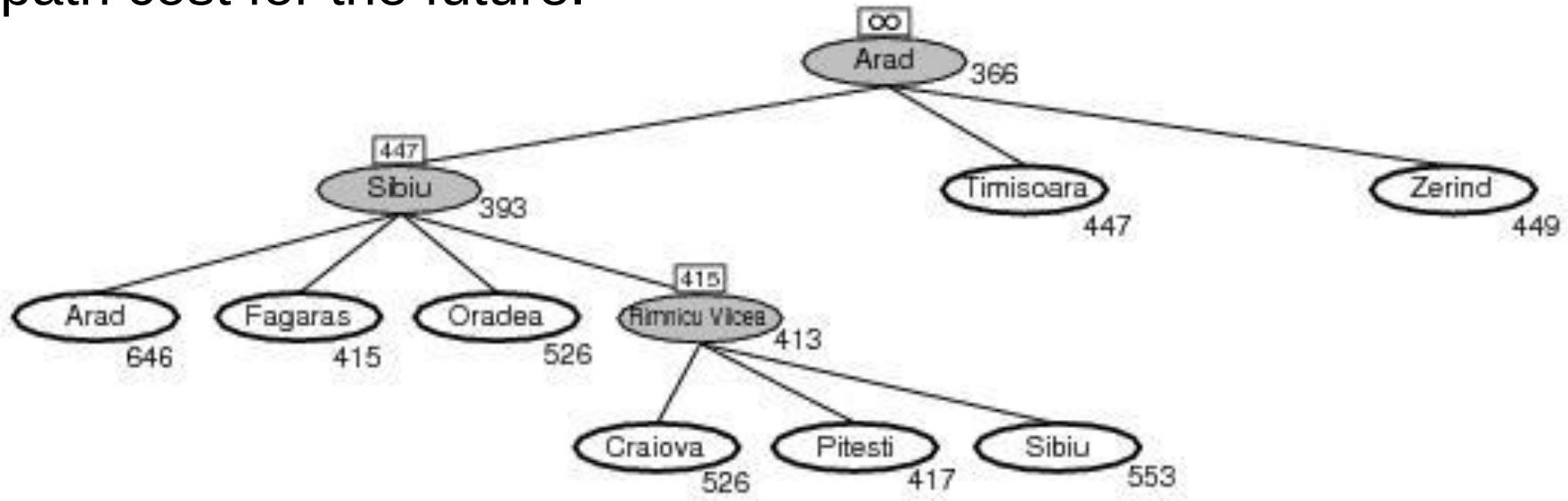
Iterative Deepening A* (IDA*)

Every iteration DF search with f-cost limit as depth limit.
Every iteration expands all nodes within the given f-cost contour and looks over the contour to find the next contour value.

IDA* **complete** and **optimal**, as A*, but memory complexity is linear.
Time complexity depends on how many values the f cost can take.

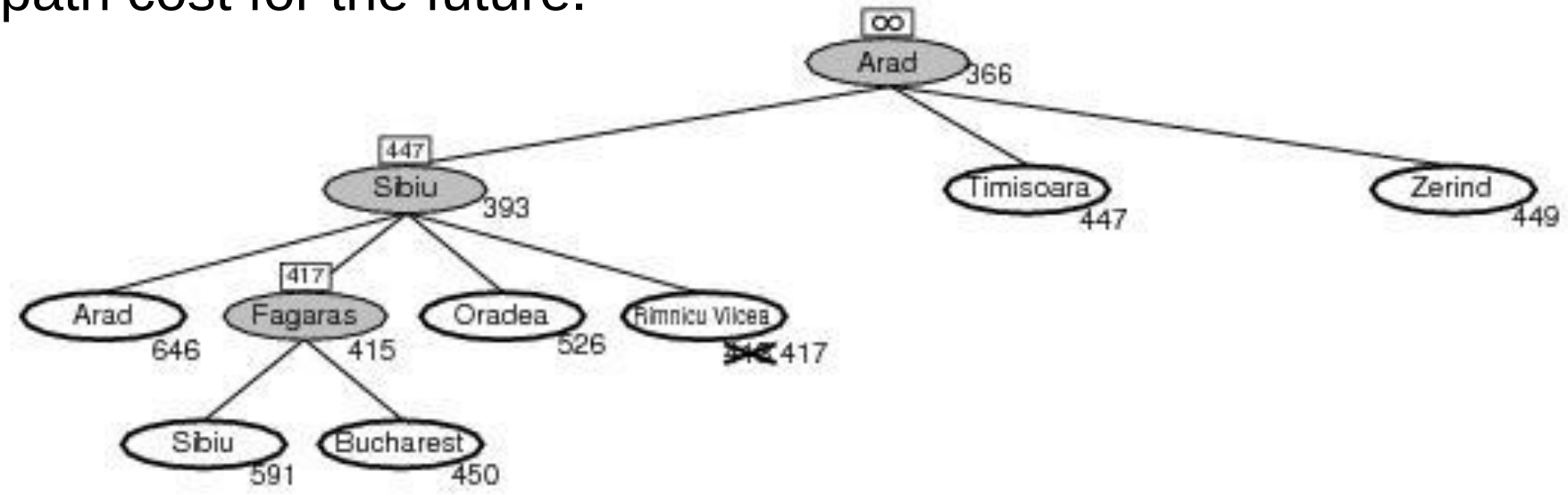
Rekursive Best-First search (Ro-RLEK)

Does BF search, acc. to f-cost, until a better alternative appears in left unexpanded branches.
Search switches over, freeing memory, but stores the actual path cost for the future.



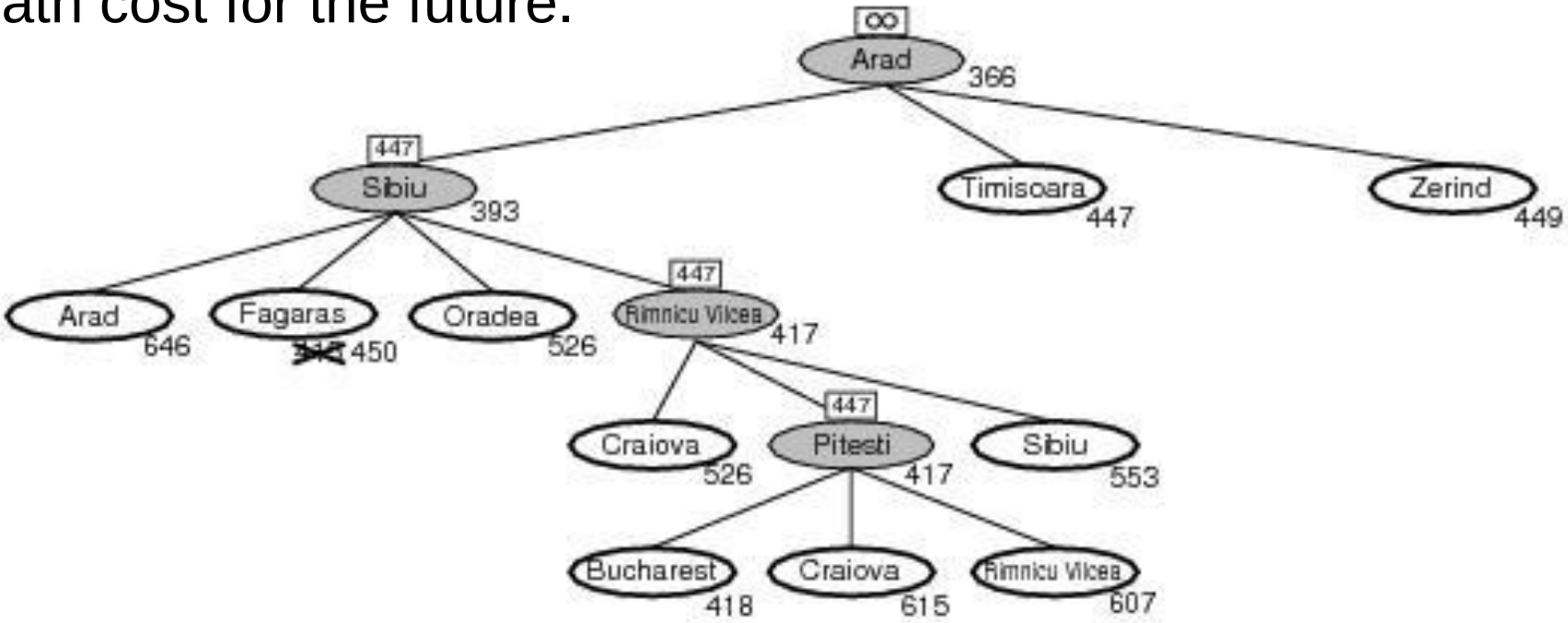
Rekursive Best-First search (Ro-RLEK)

Does BF search, acc. to f-cost, until a better alternative appears in left unexpanded branches.
Search switches over, freeing memory, but stores the actual path cost for the future.



Rekursive Best-First search (Ro-RLEK)

Does BF search, acc. to f-cost, until a better alternative appears in left unexpanded branches.
Search switches over, freeing memory, but stores the actual path cost for the future.

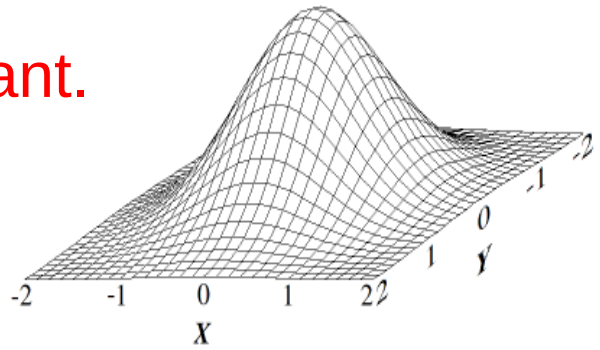


Local Search

Looking for optimum place = description of the optimal solution.
Generally only the actual state is stored.
We look for alternatives only in the vincinity of the actual state.

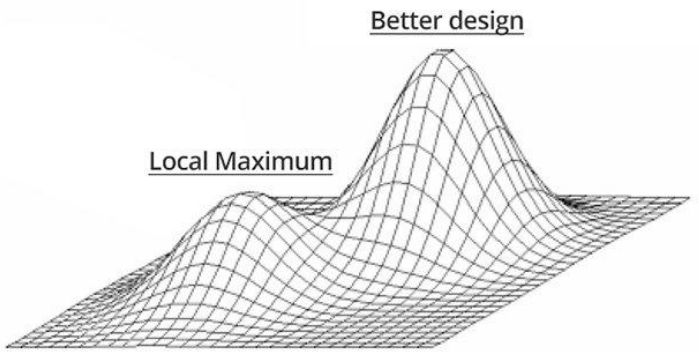
Special features of the search space ...

No backtracking, memory complexity constant.
Time complexity linear.
Solvability warranty? (complete?, optimal?)



Hill-Climbing (discrete gradient)
Simulated Annealing
Local beam search
Random start … …
Genetic algorithms
… ...

Hill-Climbing (Ro-HegyMaszo-K)



- Always tends toward better alternative
- Does not manage search tree

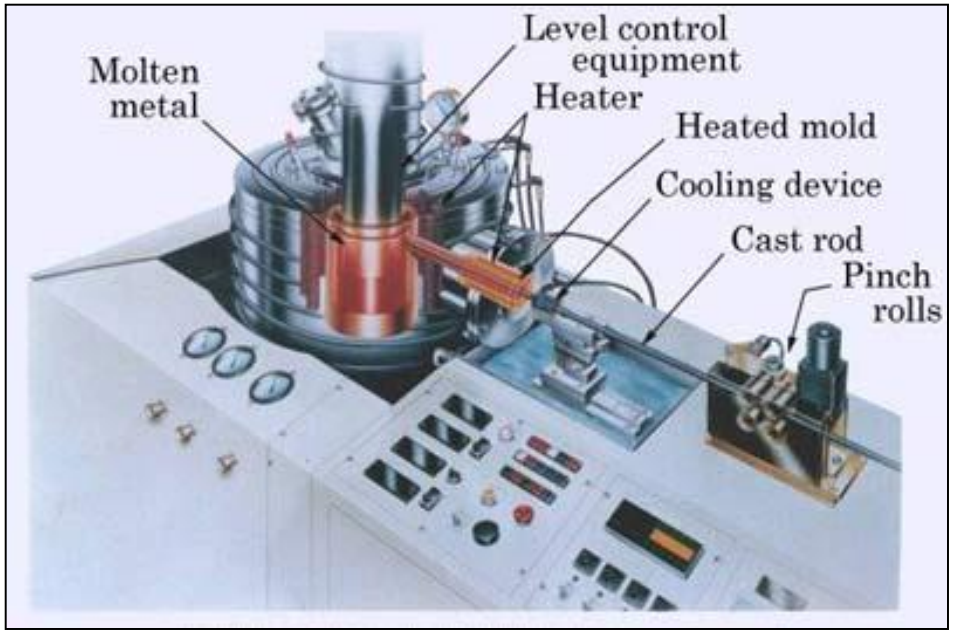
3 main problems

local maximum: stops in the first local optimum
plateau: no difference in direction of bettering, random choice
ridge: side slopes of the ridge steep, ridge slope can be very mild.

Randomly started hill-climbing

HC fast, can be frequently restarted
- time spent
- no improvement

Simulated Annealing

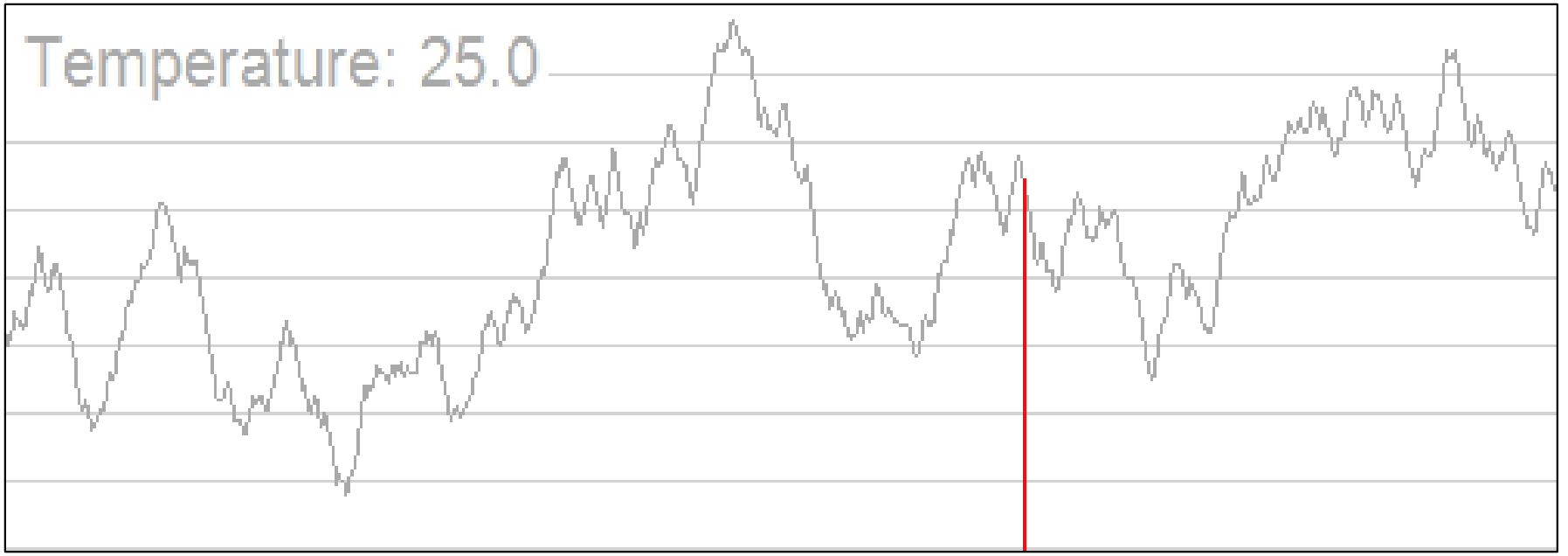


Instead of random restart
-It is permissible to go some steps back (downwards), to get out from the local optimum.

SA:
- instead of the best action a **random step** is chosen.
- if the step is an **improvement**, it is always done.
- if not, it is accepted only with a **probability** (Boltzmann-distribution).

$$P \approx \exp(-\Delta E / T)$$

The probability is exponentially decreasing with the ‘’corrupting’’ capability of the step
– ΔE = deterioration in the cost function
- T = “temperature”, cooling as the time (search steps) is going on.



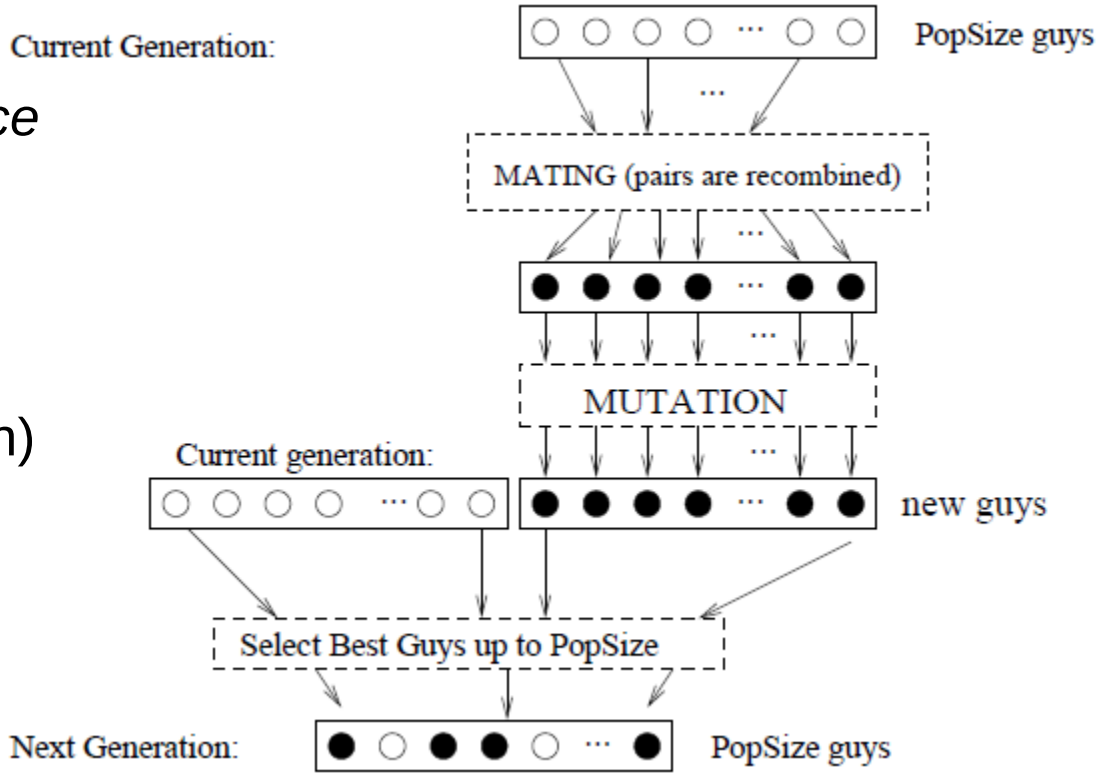
http://en.wikipedia.org/wiki/Simulated_annealing

Genetic algorithms

Start: k randomly generated state = population
Every state (individul) = defined over a finite alphabet (typically 0/1 string)
= problem coding

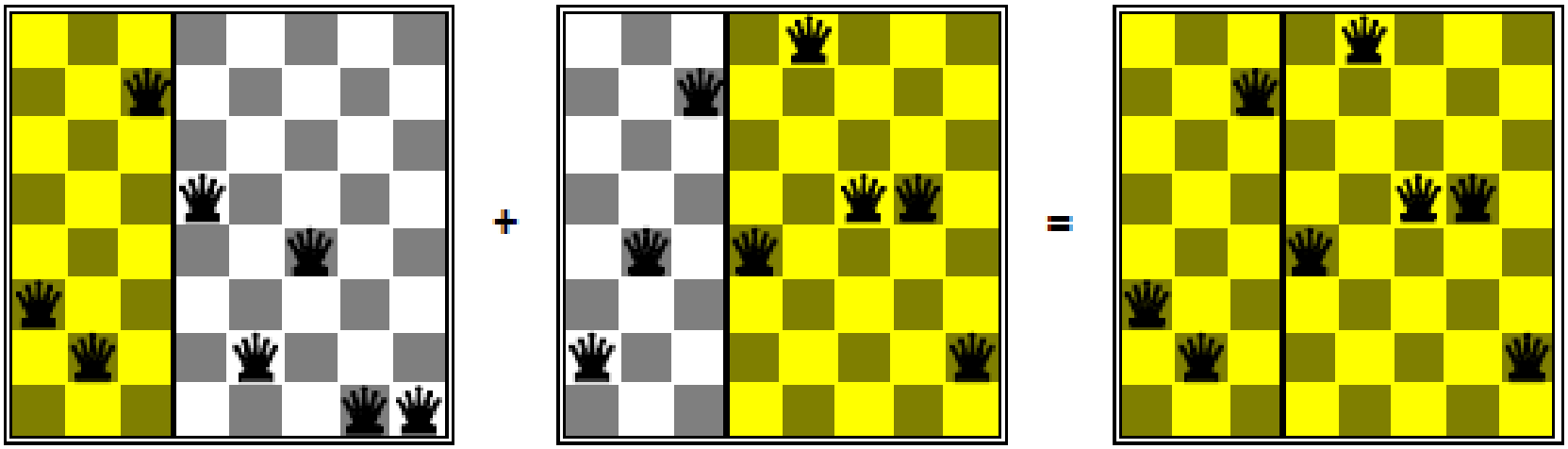
- population
- fitness-function
- cross-over (selection, pairing) depending on fitness
- mutation
- *elitism*
- *Darwin-/ Lamarck-inheritance*
- Shaping new population

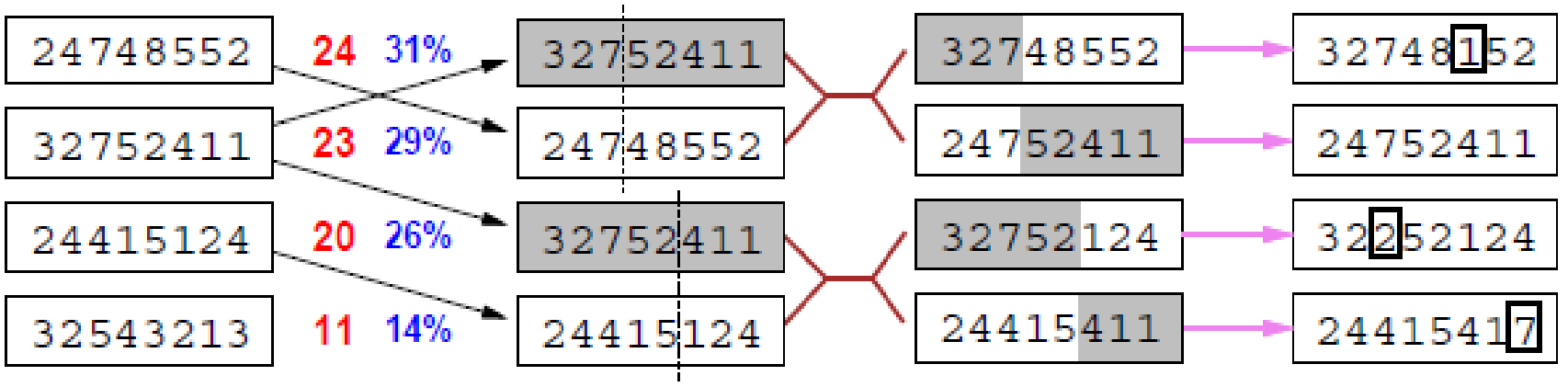
Like Hill-Climbing, etc.
(difference in the reproduction)

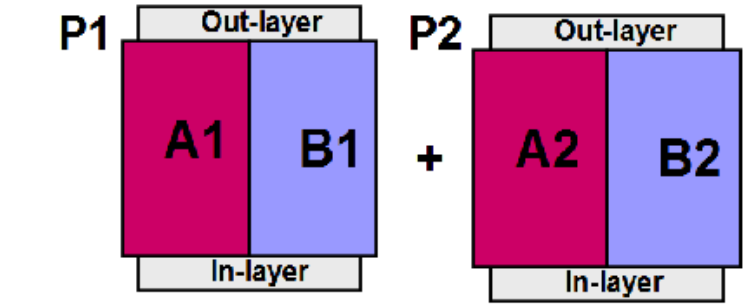


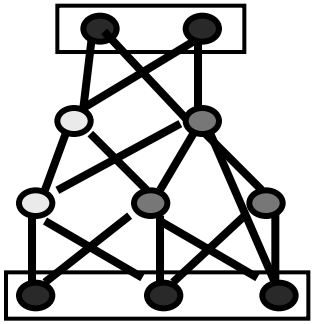
Genetic algorithms

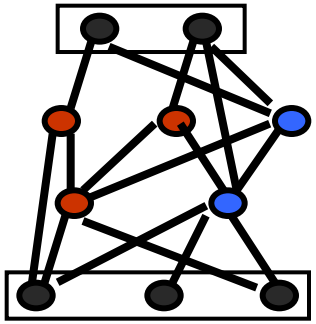
8-queen problem
fitness = 28 - n

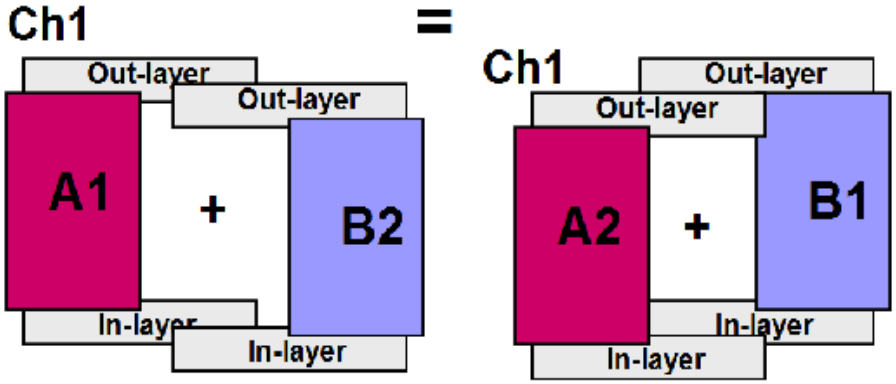


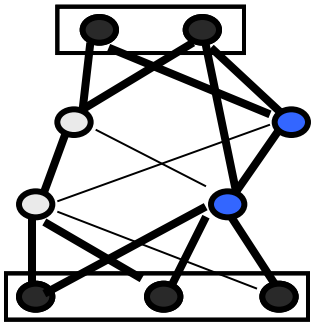


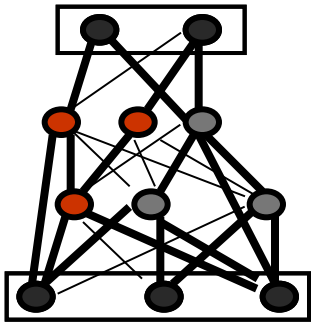












Darwin, Lamarck?

2006 NASA ST5 spacecraft antenna
https://en.wikipedia.org/wiki/Evolved_antenna

And still more and more search algorithms ...

a. Every search algorithm has plenty of special versions

b. Finding new search algorithms – always a hot AI topic

c. Gradient based procedures in continuous spaces

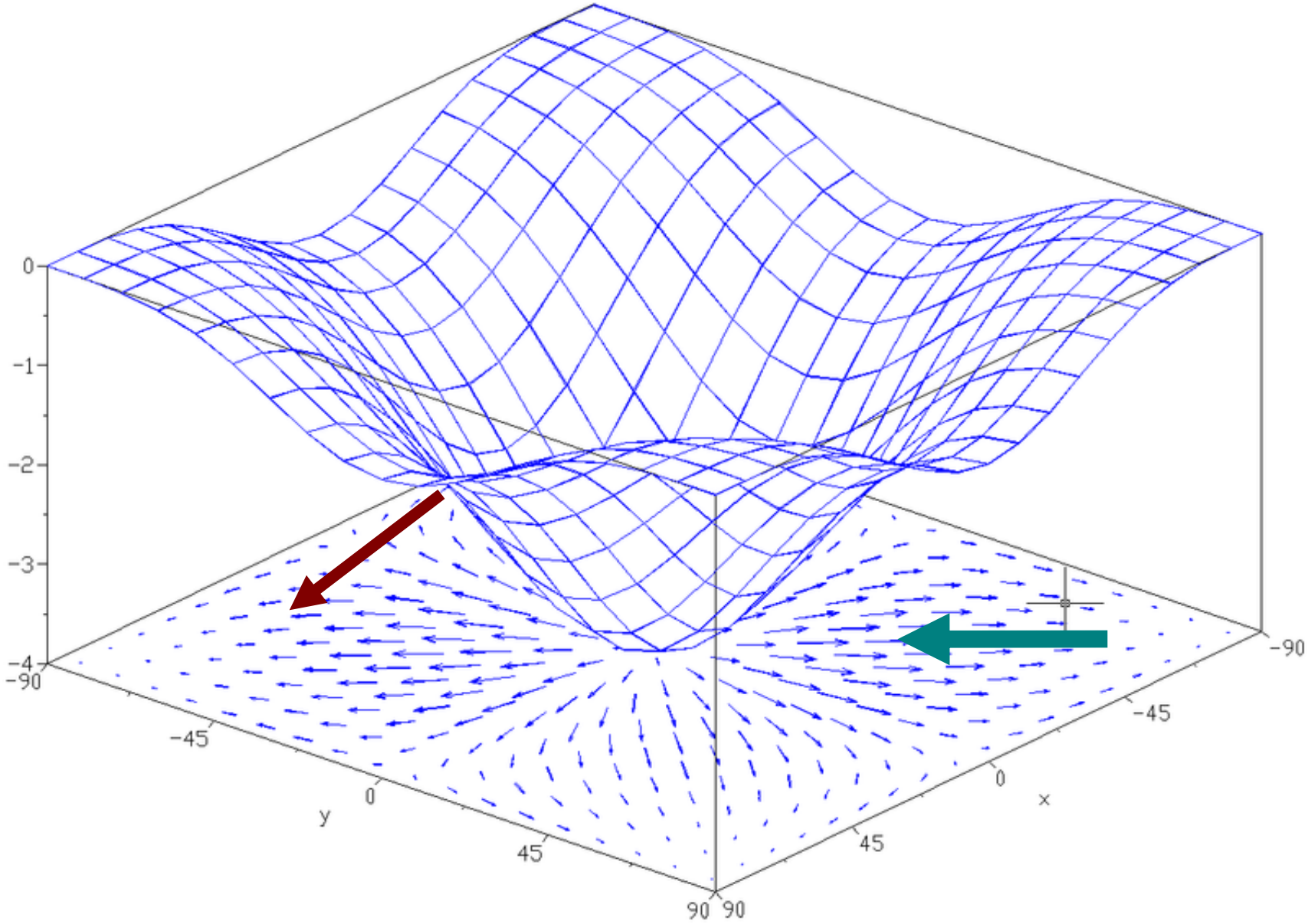
$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f(\mathbf{x})$

$\mathbf{x} \leftarrow \mathbf{x} - \alpha \mathbf{H}^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$

Relaxation method
Gradient with a fixed step
Gradient with optimized steps
Newton-method
And many more

d. (Tunelled hill-climbing (minimizing + tunnelling) ...)

e. On-line search when the information is deficient or real-time changing (exploratory problems), e.g. **learning real-time A***, etc.


0
-1
-2
-3
-4
-90
-45
0
45
90
y
-90
-45
0
45
90
x

Summary

- Heuristic function
- Admissible heuristics and A*
- Local search
- Suggested reading
  - Prieditis: Machine Discovery of Effective Admissible Heuristics, 1993

Experimenting

BF, DF, limDF, ID,
UC, BF, A*, HC, …??