AI / CH3

Informed Search

Uses problem-specific knowledge

More efficient than uninformed search

* Best-first Search:

→ Expanding due to an evaluation function $f(n)$

↓

evaluation measures the distance to the goal

↓

Node with **lowest** evaluation is selected for expansion

↓

Other key component: heurestic function $h(n)$

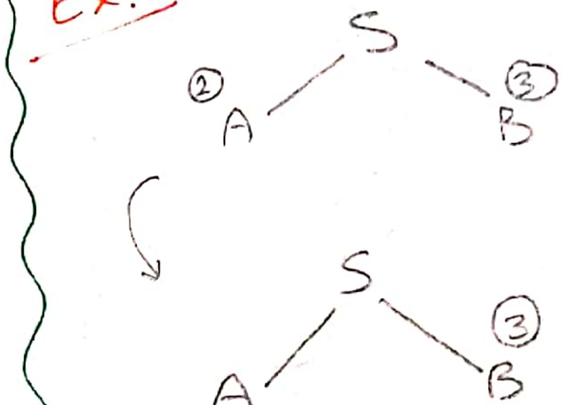
if n is a goal ⇒ $h(n)=0$

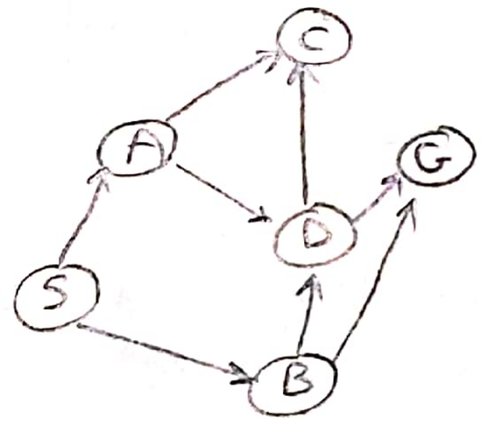
*Greedy best-first search

Tries to expand the closest node to the goal

Evaluates nodes only by the heurestic function

Ex:

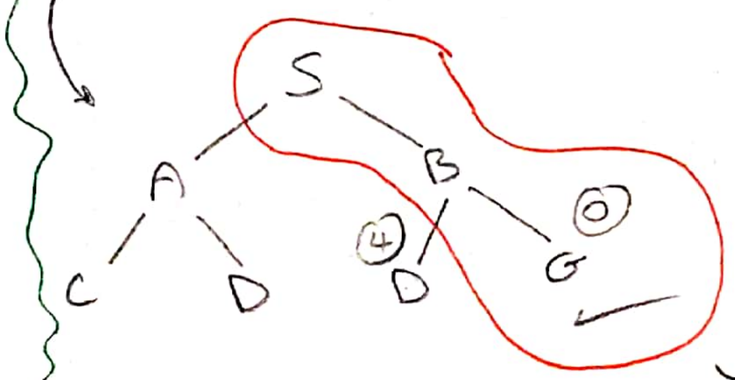




Heurestic Values:

A=2 C=1 S=10

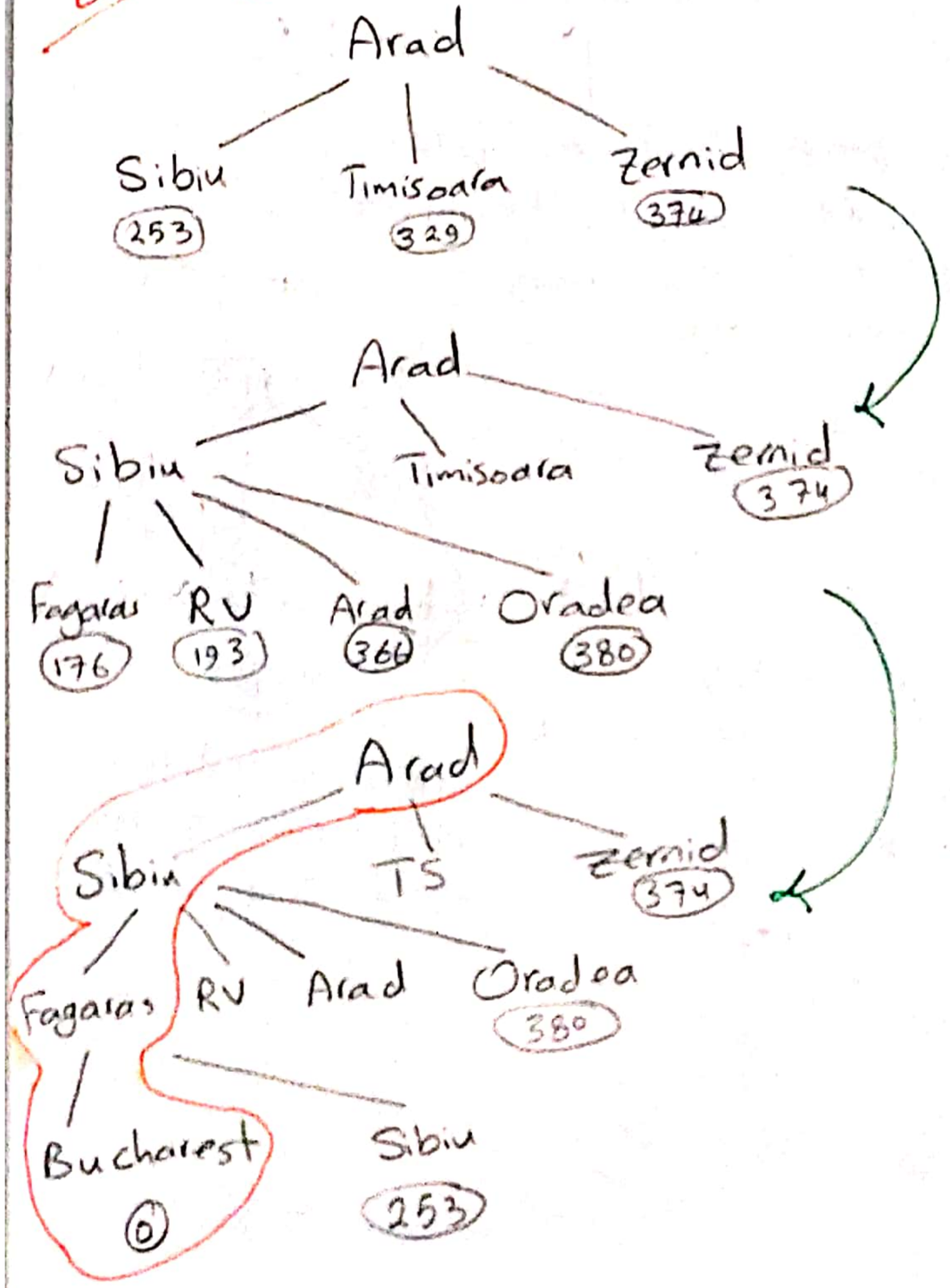
B=3 D=4 G=0



→ Path: SBG

Another Ex

Arad → Bucharest



*Properties of Greedy Best-First Search

*Complete? No

↳ can get stuch in loops

*Time? $O(b^m)$ (m: max depth, b: branching factor)

↳ Good heuristic can improve

* Space? $O(b^m)$

* Optimal? No

↳ Can stuch in infinite path

↓

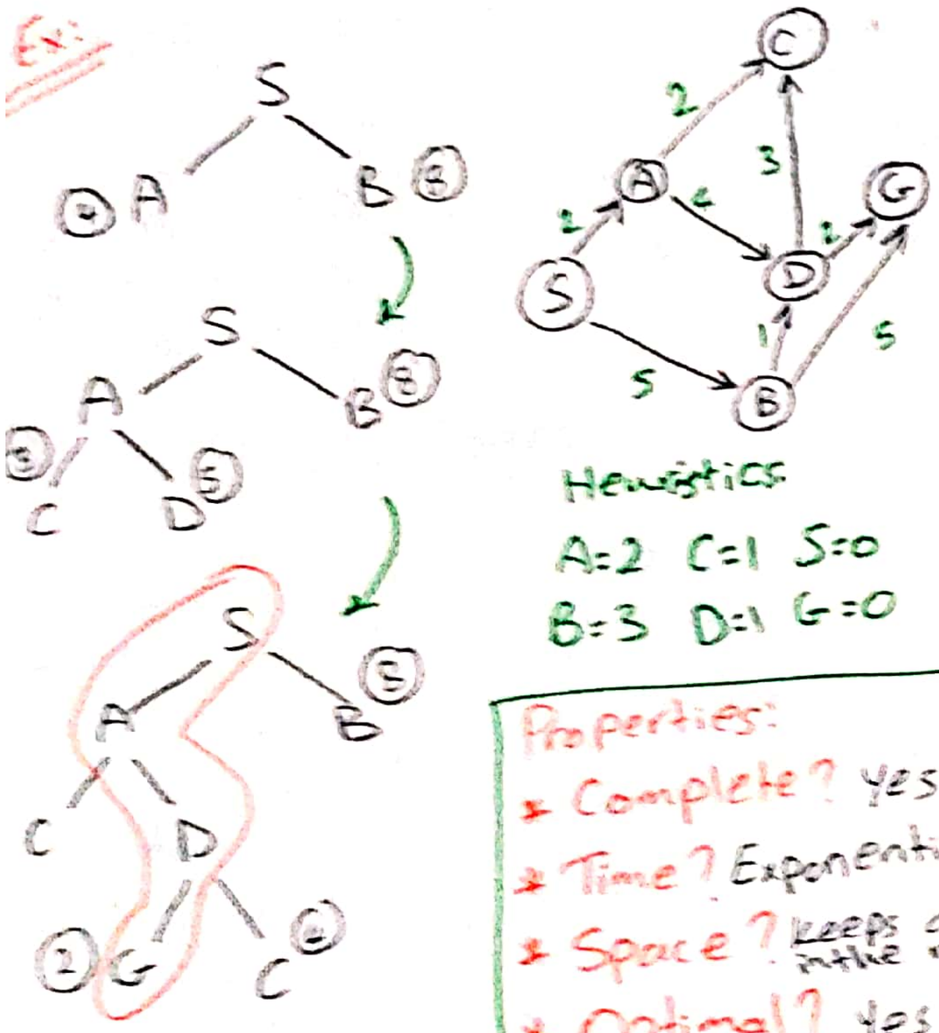
never return to try other possibilites

A* Search

$f(n) = g(n) + h(n)$

- $f(n)$: Evaluation function
- $g(n)$: Real cost
- $h(n)$: Estimated cost (heuristic)

Ex:



Heuristics

A=2 C=1 S=0

B=3 D=1 G=0

Properties:

- Complete? yes
- Time? Exponential
- Space? keeps all nodes in the memory
- Optimal? yes

Admissible heuristic

$h(n) \leq h^*(n)$ → true cost

↓

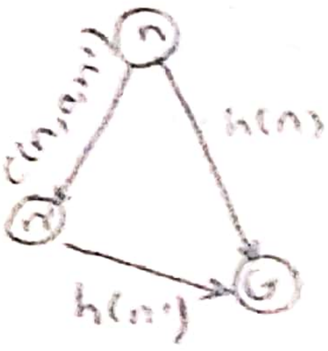
Theorem: If $h(n)$ is admissible

→ A* using Tree-Search is optimal

Consistent Heuristicts

(stricter than admissibility)

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



↓

Theorem: If $h(n)$ is consistent

→ A* using Graph-Search is optimal

Note: Uniform cost with admissble estimate is A* Search

AE) CH5

* Dominance:

$h_2(n) \geq h_1(n) \quad \forall n$

↓

h_2 dominates h_1

* Relaxed Problem (same problem, fewer restrictions)

- Fewer restrictions on actions
- Cost of optimal sol of relaxed problem is an admissble heuristic to the original problem

Kısacası: heuristic'leri üretmek için kullanılan bir yaklaşımdır.

AI Jeeto

Local Search Algorithms

- Used when the path to goal doesn't matter
- Keeps just one current (or few) state in the memory
- Good for problems have large number of states

General Approach:

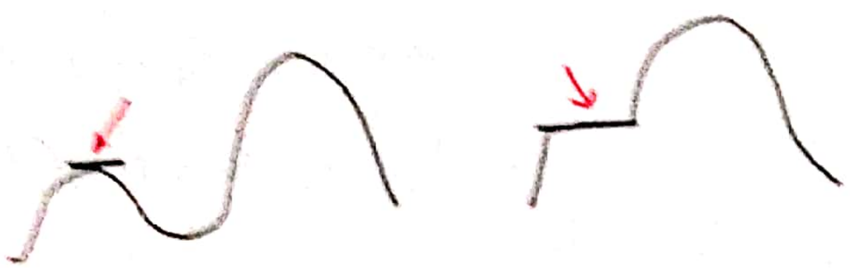
1- Generate solution
2- Test it
3- Quit or return to 1

*Hill Climbing Search (HC)

- Moves in the direction of increasing value
- Terminates when it reaches a peak

Disadvantages

- Local maximum
- Plateau



Solutions

- Moving in several directions
- Backtracking
- Big jumps

*Stochastic HC

↓

Generates successors randomly until one better than current

*Random Restart HC

↓

Generate start states randomly

↓

Proceed with hill climbing

AF) Q46

* Simulated Annealing Search (SA)

Escaping local maxima by bad moves, then gradually decrease their frequency

Move bad enough to escape local maxima but not bad enough to escape the global

Tempreture T:

$T \to 0$ Hill climbing

$T \to \infty$ Random walk

* Local Beam Search

keeps track of k state (instead of 1);

starts with k randomly generated states

generates all successors of all k states (at each iteration)

goal state? yes → Stop

goal state? no → select k best successors and repeat

AI | CH6

* Genetic Algorithms

Selection → Crossover → Mutation

- Selection → Pick 2 individuals
- Crossover → Mix two parents
- Mutation → Randomly change a location in an individual

Ex: $x_i = abcdef$

$x_1 = 435216$ (a b c d e f) $\quad x_2 = 173965$

$x_3 = 248012 \quad x_4 = 908123$

$f(x) = (a+c+e) - (b+d+f)$

1. Sorting the chromosomes by their fitness

$f(x_1) = 10 - 11 = -1$
$f(x_2) = 10 - 21 = -11$
$f(x_3) = 11 - 6 = 5$
$f(x_4) = 19 - 4 = 15$

$x_4 \rightarrow x_3 \rightarrow x_1 \rightarrow x_2$

2. Crossover

$x_3, x_4 \rightarrow$ middle

$x_3, x_1 \rightarrow$ two point (2,4)

$off_1 = 248123$
$off_2 = 908012$

$off_3 = 438016$
$off_4 = 245212$

3. fitness of offspring

$f(off_1) = 12 - 8 = 4$
$f(off_2) = 18 - 2 = 16$
$f(off_3) = 13 - 9 = 4$
$f(off_4) = 8 - 8 = 0$

AI | CH4

Constraint Satisfaction Problems (CSP)

State
↓
Defined by variables X_i with values from domain D_i

Goal state
↓
Constraints
↓
specify allowed combination of values

Consistent assignment: does not violate any constraint

Complete assignment: Every variable is mentioned

Solution: Complete assignment that satisfies all the constraints

↳ for some problems:

+ maximizing an objective function

Binary CSP: Each constraint relates two variables

For Map-Coloring Problem:

Variables: Set of cities

Domains: set of colors

Constraints: Adjacent regions must have different colors

Variaties of CSPs:

- Continuous domains
- Discrete variables
  - Finite domains
    - ↓ n variables
    - ex: map coloring
    - ex: 8 queens
  - Infinite domains
    - ↓ set of integers/strings
    - ex: Job scheduling

Variaties of Constraints

- Preferences (A is better than B)
- Unary
  - ↓ involves single variable
- Binary
  - ↓ involves pairs of variables
  - ↓ represented as constraint graph
- Higher-order
  - ↓ involves 3 or more variables

AI | CH7

* Backtracking Search → Depth first search with single var. assignment

- Cummutative
  - WA=R, NT=G
  - same ↓
  - NT=G, WA=R
- Assigns single variable at each node

* Most Constraint Variable

- To select the next unassigned variable
- Chooses the variable with fewest legal values

* Minimum Remaining Values

- Picks variable that is most likely to cause a failure soon
- strickter than MCV

* Least Constraining Value

- Chooses values that give maximum flexibility to othe variables

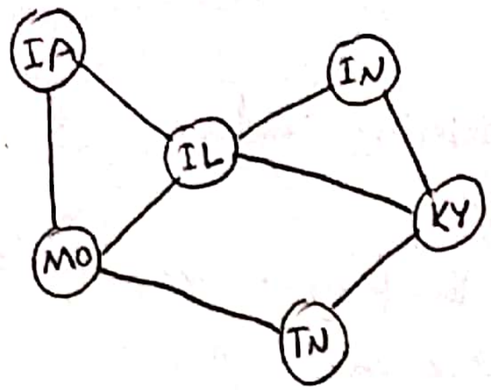
* Degree Heuristic

- Aims to reduce the branching factor by selecting variable with largest number of constraints

* Min-Conflicts Heuristic

- Illegal assignment
- Chooses value that violates the fewest constraints

a)



Forward checking

b)

TN	IN	IA	IL	MO	KY
R	RGB	RGB	RGB	~~RGB~~	~~RGB~~

Arc-consistency

c)

TN	IN	IA	IL	MO	KY
R	RGB	~~RGB~~	~~RGB~~	G	~~RGB~~

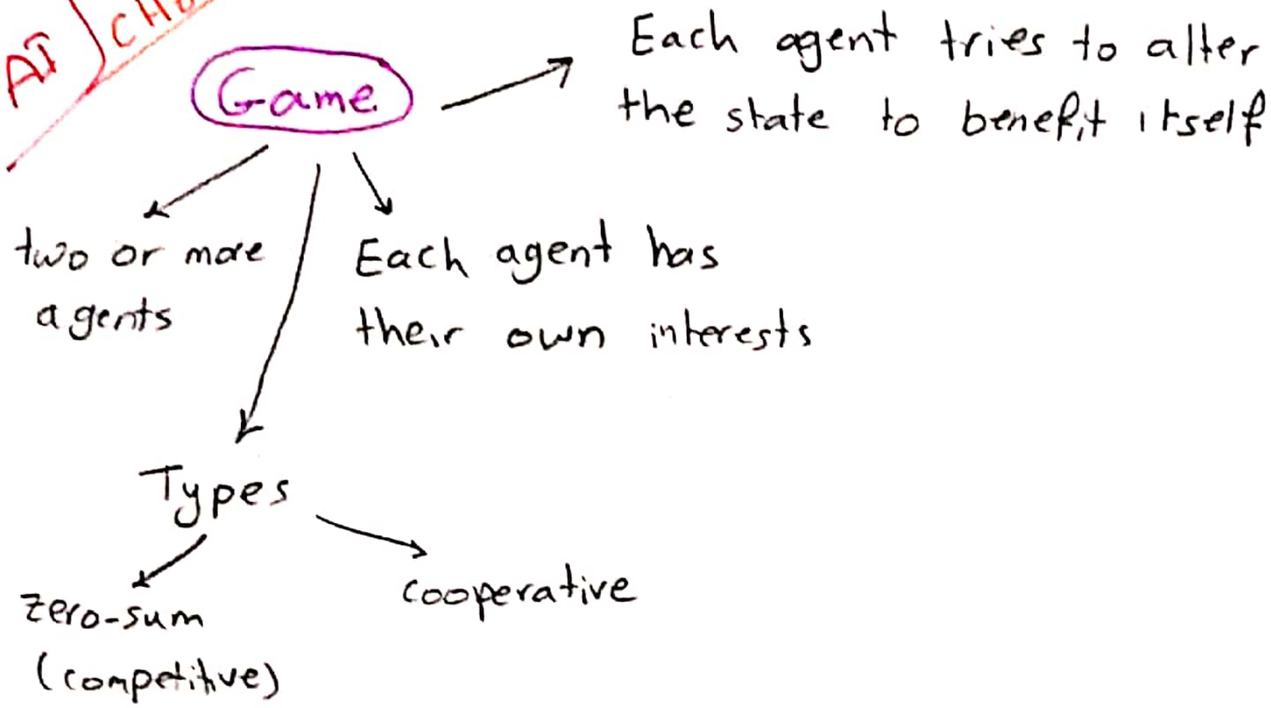
(?.)

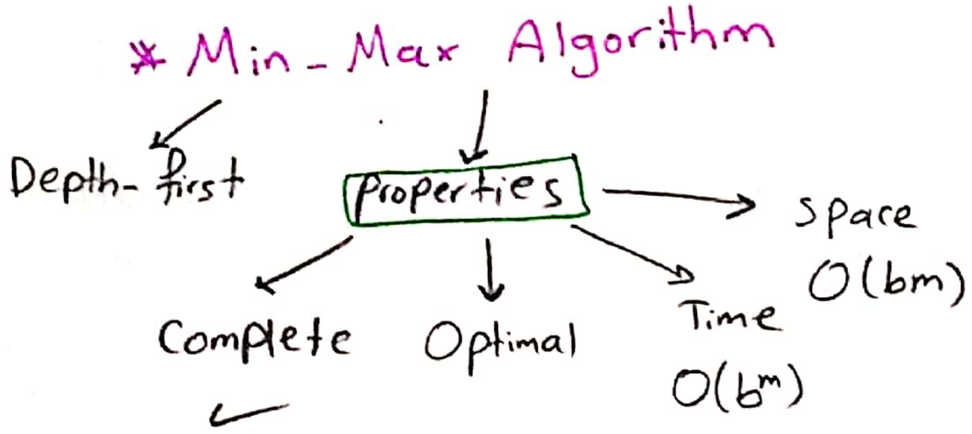
d) MRV: IL, MO

e) DH: IL

f) IA = R

AI | CH8





Asmaa Mirkhan
28.12.2019