2.Off-line local search

- Optimization: min cost function; solution: global minimum
- May fail finding a solution, even if it exists
- Simplest: Hill-climbing
- Often inspired by natural processes:
  - simulated annealing
  - genetic algorithms
  - ants algorithms

- Basically,
  - they do not keep track of states previously visited
  - look forward for immediate improvement (greediness)

1

2.Off-line local search: hill-climbing

Hill-climbing: minimize f(x)

- Generate sucessors of current state
- From x move to y such that f(x) > f(y): the move that decreases most the cost function
- Stochastic search

Local optima:

- f(x) ≤ f(w), for all w in succ(x)
- how escape from local minima?

Escape strategies:

- Accept moves to to states of higher function values with low probability (simulated annealing).

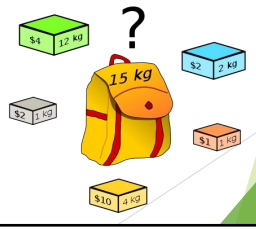
2

2.Off-line local search: genetic algorithms

Knapsack problem:

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Holland
1975



3

Genetic algorithms: Chromosome / fitness function

A candidate solution is a vector = chromosome

1	0	1	0	0

To discriminate among chromosomes:

***fitness function* F(V):** total value of vector V

F(| 1 | 0 | 1 | 0 | 0 |) = $2 + $10 = $12

4

2.Off-line local search: genetic algorithms

- Parallel search: a pool of solutions randomly generated (strings)
- Combines: exploration & exploitation

- Applying operators: generation(*i*) → offspring, generation (*i*+1)

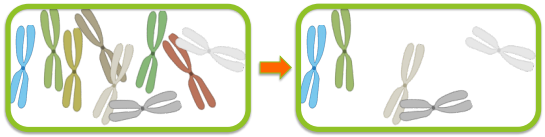
- Operators:
  - Selection : two strings are selected
  - Crossover: old strings are cut → new strings are obtained
  - Mutation: some bits may change randomly

Iterative process:
- Several generations are produced, each with better individuals
- Until a solution is found or exhaust resources

5

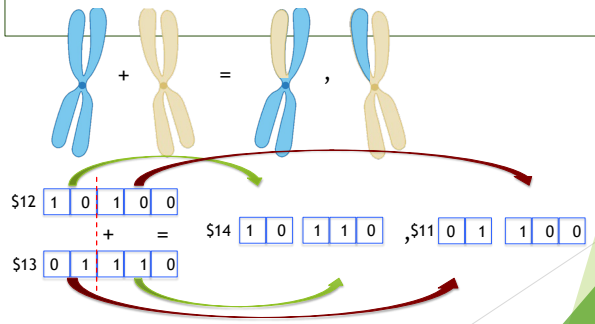
Genetic algorithms: Selection

For reproduction we select:
- The best chromosomes, according to the fitness function



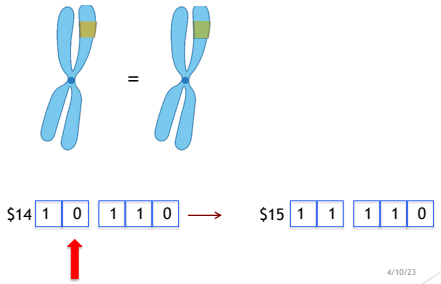
6

Genetic algorithms: Reproduction (crossover)



7

Genetic algorithms: Mutation



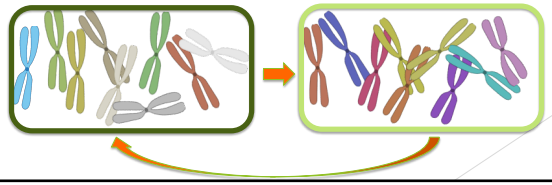
8

Genetic algorithms: New generation

Adjust the selection, crossover and mutation to obtain a new generation of K cromosomes.

The new generation replaces the old one.

The whole process repeats.



9

2.Off-line local search: suboptimality

When finding a solution:

- no guarantee that the path is optimal (= mínimum cost)

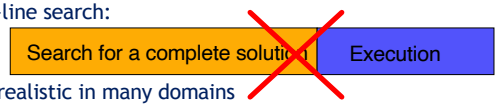
Often

- good quality solutions
- efficiency achieved

10

3.On-line search

- Off-line search:



- unrealistic in many domains

- On-line search: interleave
  - Solution computation
  - Solution execution
- Domains:
  - Unknown terrain
  - Dynamic
- Families:
  - Single-agent
  - Two agents in games, adversarial search

11

3 On-line search: single-agent

Age of Empires

Warcraft III

12

12

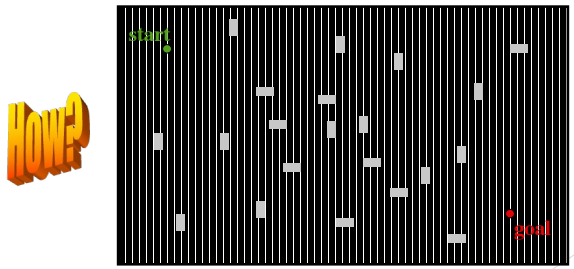
3.On-line search: single-agent approaches

- Two families:
  - Incremental search
  - Real-time search
- Basic difference: time for first move
- If time for first move could be high
  - Incremental search
- Otherwise
  - Real-time search

13

13

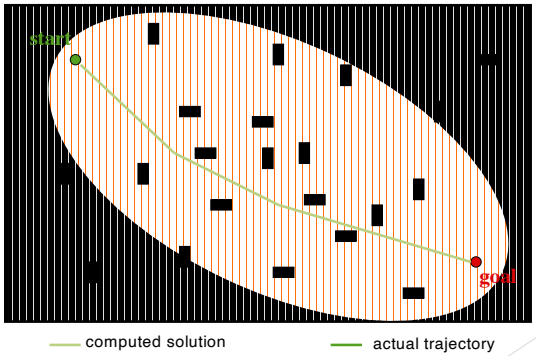
On-line search: Path-finding in unknown terrain



14

14

Off-line Search: A*



15

15

On-line search: Incremental

- As classical heuristic (off-line) search
- Suitable for **unknown environments**
- On-line search:

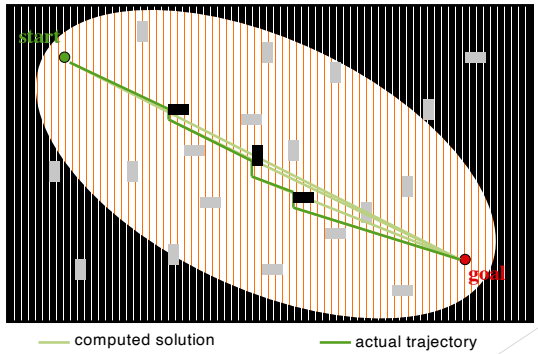
Search for a complete solution from the start state	Execution
If unfeasible, search for a new complete solution from the current state	Execution

. . . . iterates until finding a goal

16

16

On-line search: incremental, D*



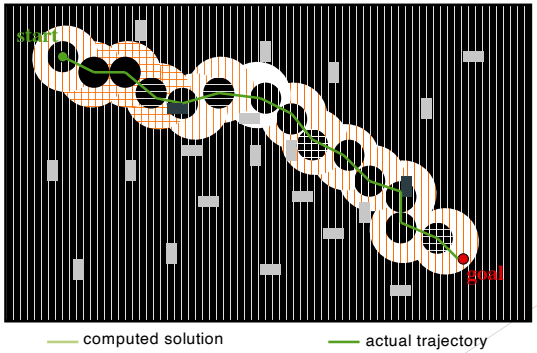
17

Real-Time Heuristic Search

- Search on a local space around the current state
- As result of search:
  - Heuristic of some states are updated (*learning space*)
  - Move(s) in the local search space
- Solution is no optimal:
  - approaches converge to optimality after repeated exec.
- Interleave search and action execution
- On-line search:

18

Real-time Search: LRTA*



19

3.On-line search. Two agents, zero sum, perfect information games: adversarial search

20

3.On-line search. Two agents, zero sum, perfect information games: adversarial search

- Two players: A and B
- Perfect information:
  - Each player knows all the information of the opponent
  - No random elements
  - *Chess, checkers, otello,go*
- Excluded:
  - Incomplete information games: *poker, bridge*
  - Stochastic games (dices): *backgammon*
- Zero-sum: if utility for A is *x* => utility for B is *-x*

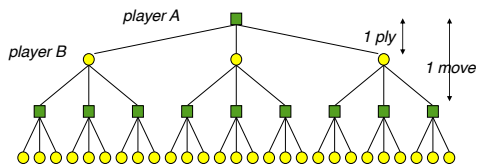
21

Game Programs

- Programs world championship: *checkers, othello*
- Program with good performance: *chess, go*
  - important: since 1956, *chess* was an IA goal
  - *Deep Blue* won *Kasparov* in 1997
  - *AlphaGo* won *Lee Sedol* in 2015
- Clear economic importance
- Strategies:
  - *brute-force*: search in spaces of billions of nodes
  - *smart* sampling, if the problem is too large

22

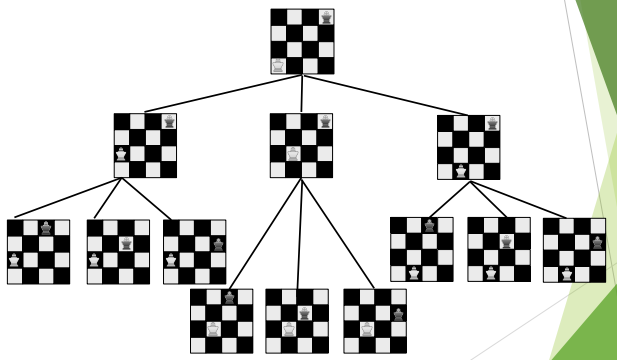
Game Tree



- Players alternate by levels
- Node successors: all legal moves that the current player can do

23

Example: Chess



24

MiniMax Main Idea

Search and back-propagate the value of the terminals (from Alan Turing, late 40's) Turing chess program

Terminal nodes: value 1, -1, 0 (A wins, B wins, draw)

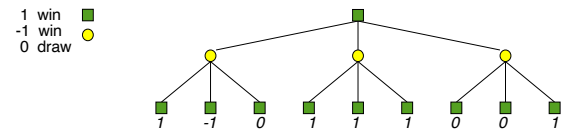
Two types of nodes: **max** wins with 1 / **min** wins with -1

Back-propagation:

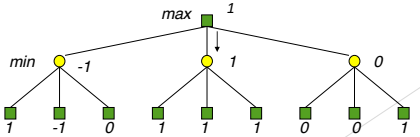
- **max**: the maximum of the values of the children
- **min**: the minimum of the values of the children

25

MiniMax Game Tree



What is the best move? Back-propagate values from node terminals, assuming that each player selects the most favourable move for him/her



EAIA

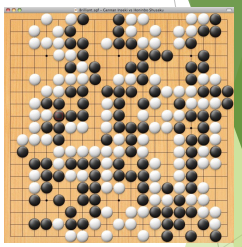
26

3.On-line adversarial search: Go

27

However, for *Go* this approach does not work!

- *Go*:
  - complex (branching~250), large (depth~150), very dynamic (no quiescence)
  - no good evaluation function
- Brute-force approaches do not work
- What about some kind of sampling? → Monte Carlo Tree Search (from 2005 on)



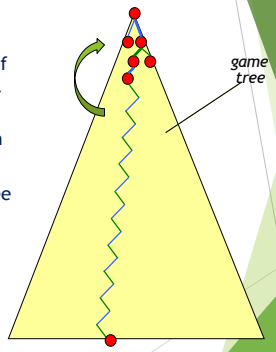
28

Monte Carlo Tree Search (MCTS)

Basic idea:

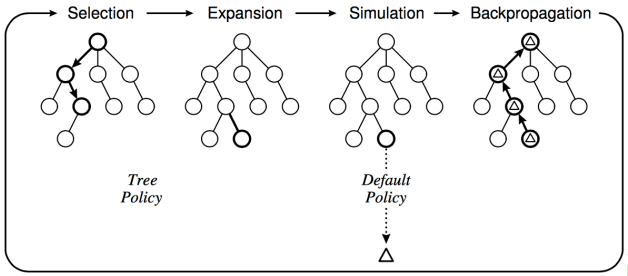
- keep (develop and store) a small part of the game tree close to the root: partial tree
- from the fringe of that partial tree, run a simulation until a terminal node
- update node statistics in the partial tree

Iterate until some limit (time or #iterations); then select the *best* move



29

MCTS: Phases



30

4.Search names: Nils Nilsson

- (1933 – 2019)
- Full profesor Stanford
- One of the big old names
- Among the creators of the A* algorithm (1968)
- Shakey robot -> STRIPS planner (1971) [*planning*]

31

4.Search names: Judea Pearl

- (1936 –)
- Full professor UCLA (retired)
- Known by his books: *Heuristics*, 1984; *Probabilistic Reasoning*, 1988; *Causality*, 2000
- Contributions to: Bayesian networks [*uncertainty*]

32

4.Search names: Richard Korf

- (195? –)
- Full profesor UCLA
- Many contributions to systematic search:
  - Off-line search: frontier search
  - On-line search: real-time heuristic search

33

33

4.Search names: Sven Koenig

- (195? –) German
- Full profesor USC (Southern California)
- Many contributions to single-agent search:
  - Partially known, non-stationary, non-deterministic domains
  - [*multi agent planning, robotics, videogames*]

34

34

4.Search names: Deep Blue

- 1997, program developed by IBM
- Mastering chess, won to the world champion Gary Kasparov
- Combination of:
  - Parallel alpha-beta search
  - Variable depth (Singular extensions)
  - Library of openings and endings

35

35

4.Search names: AlphaGo

AlphaGo

- 2015, program developed by Google DeepMind
- Mastering the Go game (more difficult than chess), won a 5-games match against the unofficial world champion Lee Sedol
- Combination of
  - Monte-Carlo Tree Search
  - Deep neural networks

36

36

5.Wrap-up (I)

- Basics:
  - A state
  - State-space of a problem
  - Successors of a state
  - Path-finding
- Off-line search
  - Systematic search
  - Local search:
    - hill climbing [local optima, global optimum]
    - genetic algorithms:
      - chromosome, fitness function
      - selection, crossover, mutation

37

37

5.Wrap-up (II)

- On-line search
  - Single agent search
    - Incremental search
    - Real-time search
  - Two-agent, adversarial search
    - Game tree
    - Games: chess, go
- Search names: Nilsson, Pearl, Korf, Koenig, Deep Blue, Alpha Go
- Algorithms mentioned: DFS, BFS, A*, genetic, MiniMax
- Examples: 8-puzzle, knapsack

38

38

Further reading

- Russel & Norvig 3rd ed:
  3.1, 3.2, 3.3, 4.1, 5.1, 5.2, 5.7 plus
  Bibliographical Notes chapters 3, 4 and 5
- Heuristics (Pearl, 1984)
- Korf's talk at ICAPS 2018 (very instructive, no local search)
  https://www.youtube.com/watch?v=X6qCBcubZIE

39

39