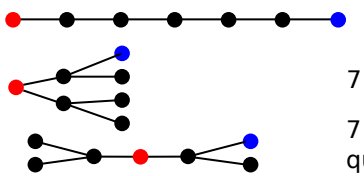
Exploiting Search Space Structure in Classical Planning: Analyses and Algorithms

Masataro Asai, Graduate School of Arts and Sciences,
The University of Tokyo, final year

1. Background

・The lack of a simple quantitative measure for describing the behavior of an algorithm caused a bunch of "new" algorithms (incl. mine)
・A*, A* with various tiebreaking, IDA*, GBFS, MCTS, MRW, DBFS, LS, Type-GBFS, BIP, BWFS, RRT
・Even if the number of expanded nodes are the same, their results can be completely different
・Number of expanded nodes, branching factor, etc., are not sufficient



7 nodes, b=1

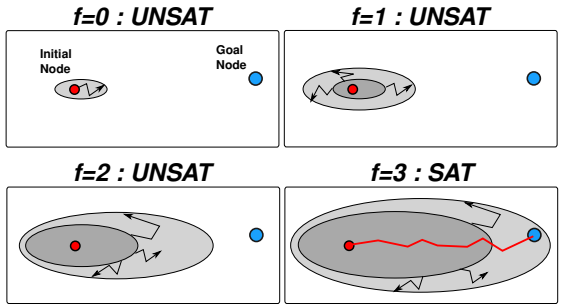
7 nodes, b=2

7 nodes, b=2, but is qualitatively different

・Is a continuum of search algorithms possible?
・To simplify the discussion, we first decompose various heuristic search algorithms into blind search

2. Optimal A* as a sequence of satisficing search

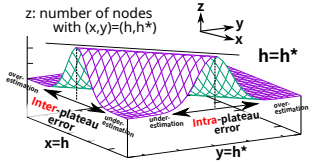
・As long as the first sorting criterion is admissible, the rest of tiebreaking does not affect the solution optimality of A*
・The search order inside a plateau produced by an admissible f does not matter
・The search inside a plateau can be **any complete**



→Now we no longer have to care about optimal search!

3. Diversified satisficing search as a combination of blind search strategies

[ICAPS17]



Visit my ICAPS17 talk & poster

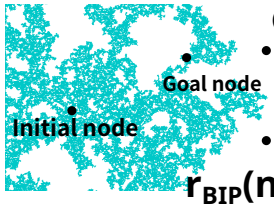
Inter-plateau Diversification: Randomized h-selection

Intra-plateau Diversification: Randomized tiebreaking

Both are knowledge-free, randomized blind search
Satisficing search = heuristics + blind search
→ We no longer have to care about heuristic search!

4. Blind search strategies

Invasion Percolation (IP) [ICAPS17]



(Wilkinsonand Willemsen '83)

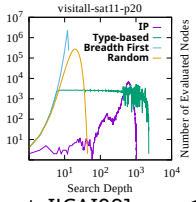
- Physical model for the distribution of fluid slowly invading porous media (e.g. water replacing the oil in a porous rock for retrieving oil)
- For details plz visit my ICAPS talk

$r_{BIP}(n) = \min_{\forall e \in parents(n)} memo(e, random())$

- Sort the search nodes according to r_{BIP} and run best-first search
- r_{BIP}: minimum of the random values memoised on each search edge
- Embankment effect: high-r_{BIP} value nodes surround a certain region and prevent / delay the exploration inside it

Type-buckets using search depth information (g,h,d) (Xie AAAI14, Asai AAAI16)

・Store the nodes in buckets labeled by the key value <g,h> or <d>
・Randomly select a node from a random bucket
・Balancing the depth distribution

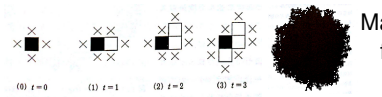


Monte-Carlo Random Walk (MRW) [Nakhost, IJCAI09]
Performs a random walk local search.
Iterative Width [Lipovetzky, ECAI12]
Tries to finds the novel states using the state information
Monte-carlo tree search (for games)
Backprop the win-ratio estimated by playouts (local search)
Expand the most promising node or by UCB

5. Randomized blind search as random fractals

Some fractals are defined by generative rules

Eden Model

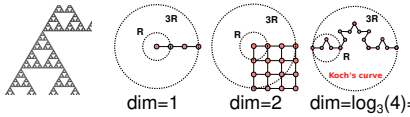


Math model for a cell culture

Nodes are randomly generated on the surface of the cluster
= Random expansion from the OPEN list

Key idea: Applying **the existing mathematical results on random fractals** to the search algorithms.

Measuring the Fractal dimension of a graph



e.g. Radius method
・Change the radius
・Count the nodes

Fractional dimensions = "sparse" objects

Sparse search yields fast exploration.

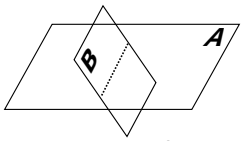
2D Invasion Percolation: $dim_H = 91/48 \sim 1.89$ [Sahimi 94]
Random walk: $dim_H = 2$ for any dimension > 4
2D Self-avoiding random Walk: $dim_H = 1.55$

Minimum fractal dimension for a search algorithm to find a solution

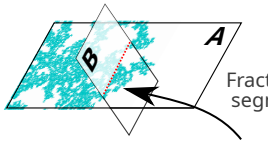
[Falconeer 1989]

Intersections of fractals *tends to be* fractals of

$$Dim(F_1 \cap F_2) = \max\{0, Dim(F_1)+Dim(F_2) - n\}$$



A ∩ B (D=2) is a line (D=1)



A(D=1.89) ∩ B (D=2) : D=0.89

・Goal states form a hypercube G in the search space
・Search algorithm needs to explore the space which has non-empty intersection with G
・Worst case: The intersection has dimension 0 (i.e. the intersection is points; very small)
・Minimum fractal dimension of the explored states of a search algorithm in a particular problem with **v** state variables: Dim(algorithm) > v - Dim(G) (speculation)

Future work:
Iterative Fractal Dimension Search