An Analytical Approach to the BFS vs. DFS Algorithm Selection Problem[1]

Tom Everitt Marcus Hutter

Australian National University

September 3, 2015

- Everitt, T. and Hutter, M. (2015a). Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search. In *28th Australian Joint Conference on Artificial Intelligence*
- Everitt, T. and Hutter, M. (2015b). Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part II: Graph Search. In *28th Australian Joint Conference on Artificial Intelligence*

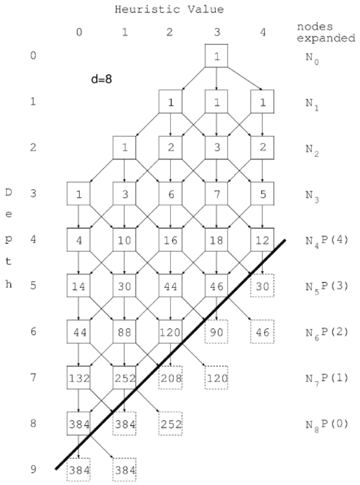
[1] BFS=Breadth-first search, DFS=Depth-first search

Outline

Motivation

- (Graph) search is a fundamental AI problem:
  planning, learning, problem solving
- Hundreds of algorithms have been developed, including metaheuristics such as simulated annealing, genetic algorithms.
- These are often heuristically motivated, lacking solid theoretical footing.
- For theoretical approach, return to basics: BFS and DFS.
- So far, mainly worst-case results have been available (we focus on average/expected runtime).

Breadth-first Search (BFS)



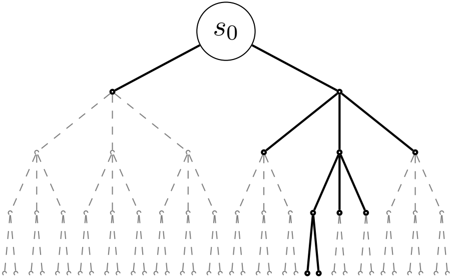
Korf et al. (2001) found a clever way to analyse IDA*, which essentially is a generalisation of BFS.

Later generalised by Zahavi et al. (2010).

Both are essentially worst-case results.

Depth-first Search (DFS)

Knuth (1975) developed a way to estimate search tree size and DFS worst-case performance.



Assume the same number of children in other branches.

Estimate $\approx 2 \cdot 3 \cdot 3 \cdot 2 = 36$ leaves.

Refinements and applications

- Purdom (1978): Use several branches instead of one
- Chen (1992): Use *stratified* sampling
- Kilby et al. (2006): The estimates can be used to select best SAT algorithm

Potential gains

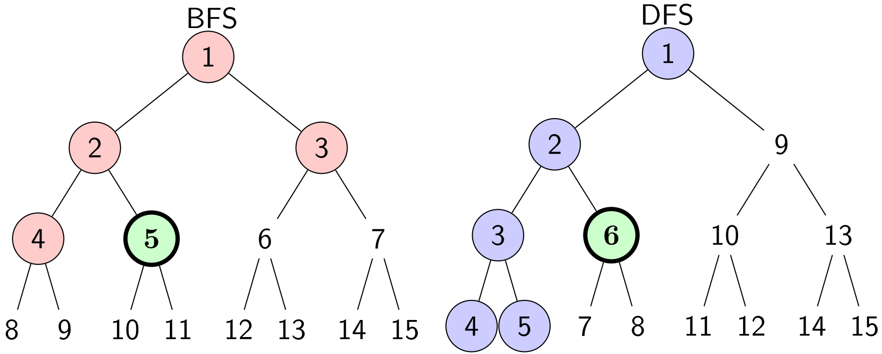
We focus on average or expected runtime of BFS and DFS rather than worst-case.

Selling points:

- Good to have an idea how long a search might take
- Useful for *algorithm selection* (Rice, 1975)
- May be used for constructing meta-heuristics
- Precise understanding of basics often useful

BFS and DFS

BFS and DFS are opposites.



focuses near the start node

focuses far from the start node

Formal setups

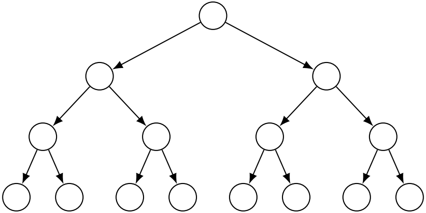
We analyse BFS and DFS expected runtime in a sequence of increasingly general models.

1. Tree with a single level of goals
2. Tree with multiple levels of goals
3. General graph

Increasingly coarse approximations are required

Simplest model - Tree with Single Goal Level

Our simplest model assumes a complete tree with:

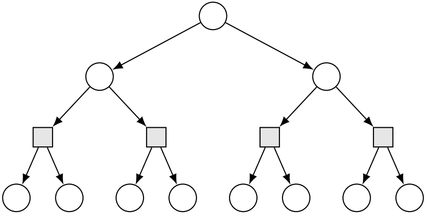


$D = 3$,

- A max search depth $D \in \mathbb{N}$,

Simplest model - Tree with Single Goal Level

Our simplest model assumes a complete tree with:

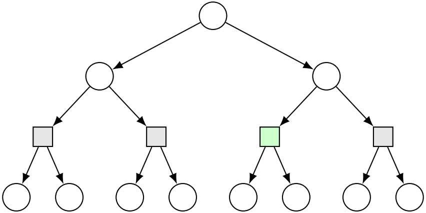


$D = 3,\ g = 2,$

- A max search depth $D \in \mathbb{N}$,
- A goal level $g \in \{0, \ldots, D\}$

Simplest model - Tree with Single Goal Level

Our simplest model assumes a complete tree with:

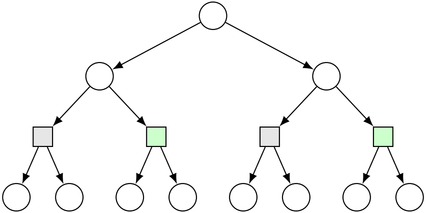


$D = 3$, $g = 2$, $p = 1/3$

- A max search depth $D \in \mathbb{N}$,
- A goal level $g \in \{0, \ldots, D\}$
- Nodes on level g are goals with goal probability $p \in [0, 1]$ (iid).

Simplest model - Tree with Single Goal Level

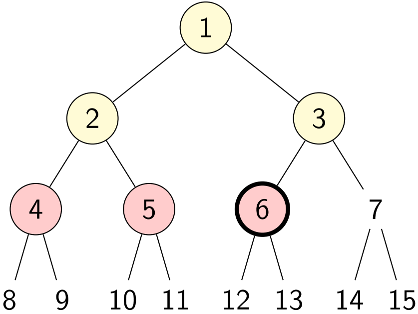
Our simplest model assumes a complete tree with:



$D = 3$, $g = 2$, $p = 1/3$

- A max search depth $D \in \mathbb{N}$,
- A goal level $g \in \{0, \ldots, D\}$
- Nodes on level g are goals with goal probability $p \in [0, 1]$ (iid).

BFS Runtime

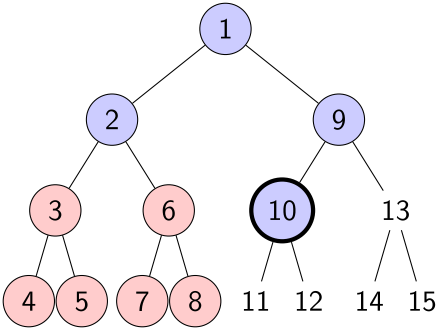


Expected BFS search time is

$$\mathbb{E}[t_{\mathrm{BFS}}] = 2^g - 1 + 1/p$$

Proof. The position Y of the first goal is geometrically distributed with $\mathbb{E}[Y] = 1/p$.

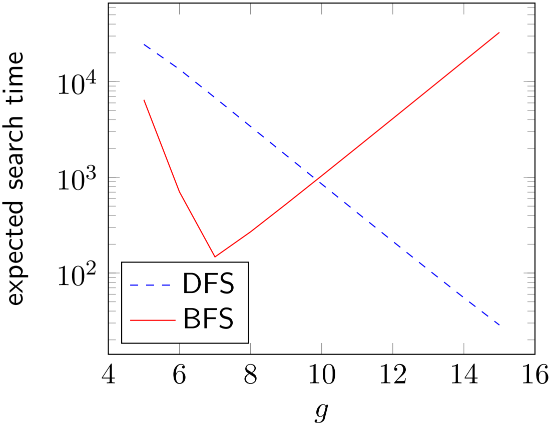
DFS Runtime



Expected DFS search time is

$$\mathbb{E}[t_{\mathrm{DFS}}] \approx \underbrace{(1/p-1)}_{\text{number of subtrees}} \underbrace{2^{D-g+1}}_{\text{size of subtrees}}$$

Proof. There are $(1/p-1)$ red mini-trees of size $\approx 2^{D-g+1}$. It turns out that the blue nodes do not substantially affect the count in most cases.



Expected BFS and DFS search time as a function of goal depth in a tree of depth $D = 15$, and goal probability $p = 0.07$.
The initially high expectation of BFS is because likely no goal exists $\rightsquigarrow$ whole tree searched (artefact of model).

Combining the runtime estimates yields an elegant decision boundary for when BFS is better:

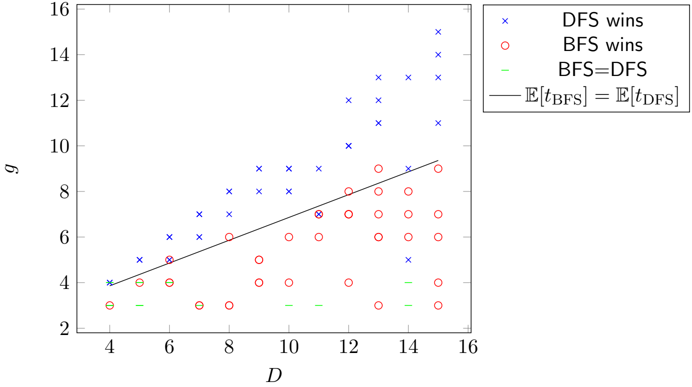
$$\underbrace{\mathbb{E}[t_{\mathrm{BFS}}] - \mathbb{E}[t_{\mathrm{DFS}}] < 0}_{\text{BFS Better}} \iff g < D/2 + \gamma$$

where $\gamma = \log_2(\frac{1-p}{p})/2$ is inversely related to p
(γ small when p not very close to 0 or 1).

Observations:

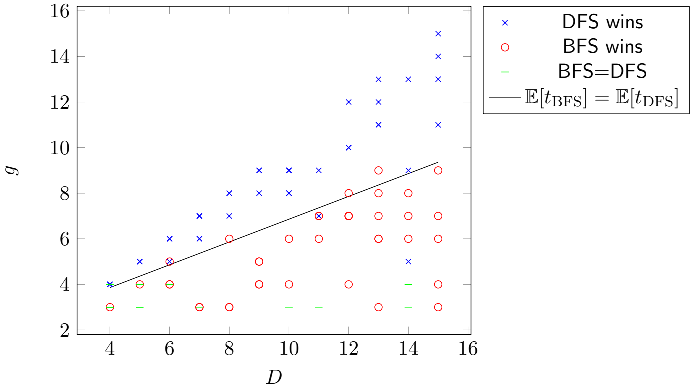
- BFS is better when goal near start node (expected)
- DFS benefits when p is large

BFS vs. DFS



Plot of BFS vs. DFS decision boundary with goal level g and goal probability $p = 0.07$. The decision boundary gets 79% of the winners correct.

BFS vs. DFS

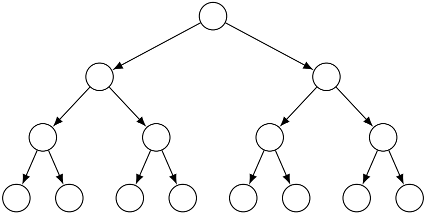


Plot of BFS vs. DFS decision boundary with goal level g and goal probability $p = 0.07$. The decision boundary gets 79% of the winners correct.

Time to generalise.

Tree with Multiple Goal Levels

As before, assume a complete tree with:

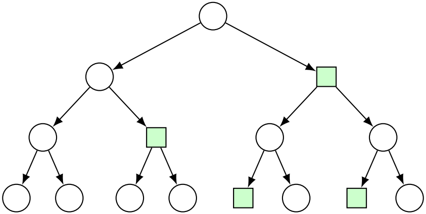


$D = 3,$

- A maximum search depth D

Tree with Multiple Goal Levels

As before, assume a complete tree with:



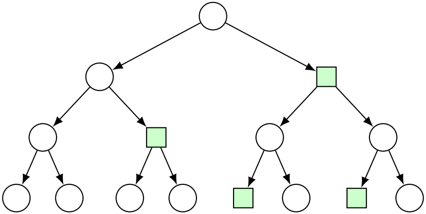
$D = 3$, $\mathbf{p} = [0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

- A maximum search depth D
- Instead of goal level g and goal probability p:
  Use a goal probability vector $\mathbf{p} = [p_0, \ldots, p_D]$. Nodes on level k are goals with iid probability p_k.

This is arguably much more realistic :) ways to estimate the goal probabilities is an important future question.

Tree with Multiple Goal Levels

As before, assume a complete tree with:



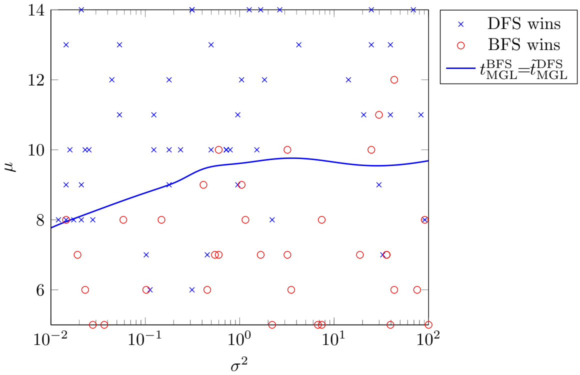
$D = 3$, $\mathbf{p} = [0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

- A maximum search depth D
- Instead of goal level g and goal probability p:
  Use a goal probability vector $\mathbf{p} = [p_0, \dots, p_D]$. Nodes on level k are goals with iid probability p_k.

This is arguably much more realistic :) ways to estimate the goal probabilities is an important future question.

- Both BFS and DFS analysis can be carried back to the single goal level case with some hacks.
- BFS analysis is fairly straightforward
- DFS requires approximation of geometric distribution with exponential distribution

Decision Boundary

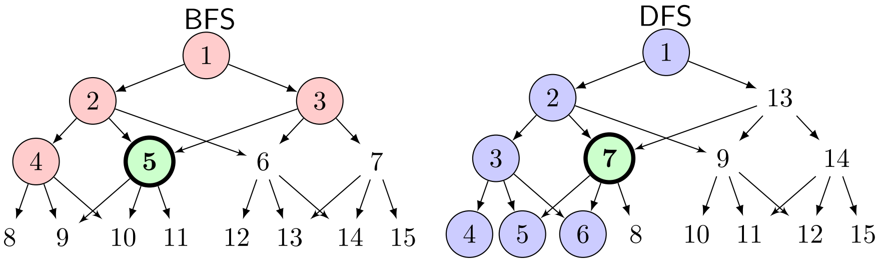


The goal probabilities are highest at a peak level μ, and decays around it depending on σ^2.

Some takeaways:

- BFS still likes goals close to the root
- BFS likes larger spread more than DFS does (increases probability of really easy goal)

General graphs

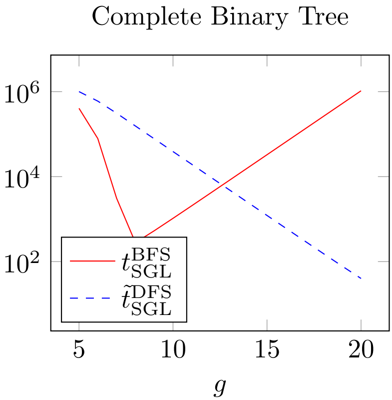


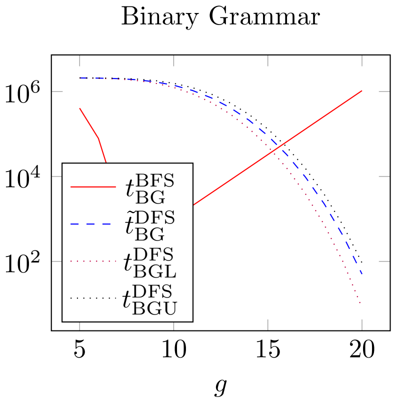
We capture the various topological properties of graphs in a collection of parameters called the descendant counter.

Similarly to before, we get approximate expressions for BFS and DFS expected runtime given a goal probability vector.

We analytically derive the descendant counter for two concrete grammar problems (it could potentially be inferred empirically in other cases).

One observation is that DFS can spend an even greater fraction of the initial search time far away from the root.





So BFS will be better for a wider range of goal levels in graph search than in tree search.

Experimental results

We randomly generate graphs according to a wide range of parameter settings.

BFS always accurate.

DFS in trees:
Usually within 10% error; in some corner cases up to 50% error.

DFS in binary grammar problem (non-tree graph):
Mostly within 20% error; 35% at worst.

More detailed results in paper.

Conclusions

With our model of goal distribution, we can predict *expected* search time of BFS and DFS (instead of only worst-case), given goal probabilities for all distances.

Further work needed to automatically infer parameters.

This theoretical understanding can hopefully be useful when:

- Choosing search method
- Constructing meta-heuristics
- Analysing performance of more complex search algorithms (for example, A* is a generalisation of BFS, and Beam Search is a generalisation of DFS).
- Choosing graph representation of search problem.

References

Chen, P. C. (1992). Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs. *SIAM Journal on Computing*, 21(2):295–315.

Everitt, T. and Hutter, M. (2015a). Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search. In *28th Australian Joint Conference on Artificial Intelligence*.

Everitt, T. and Hutter, M. (2015b). Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part II: Graph Search. In *28th Australian Joint Conference on Artificial Intelligence*.

Kilby, P., Slaney, J., Thiébaux, S., and Walsh, T. (2006). Estimating Search Tree Size. In *Proc. of the 21st National Conf. of Artificial Intelligence, AAAI, Menlo Park.*

Knuth, D. E. (1975). Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29(129):122–122.

Korf, R. E., Reid, M., and Edelkamp, S. (2001). Time complexity of iterative-deepening-A*. *Artificial Intelligence*, 129(1-2):199–218.

Purdom, P. W. (1978). Tree Size by Partial Backtracking. *SIAM Journal on Computing*, 7(4):481–491.

Rice, J. R. (1975). The algorithm selection problem. *Advances in Computers*, 15:65–117.

Zahavi, U., Felner, A., Burch, N., and Holte, R. C. (2010). Predicting the performance of IDA* using conditional distributions. *Journal of Artificial Intelligence Research*, 37:41–83.