Optimal Prefetching in Random Trees

Alain Jean-Marie

Inria – University of Montpellier

Kausthub Keshava

Deloitte (formerly at IISER Mohali)

Sara Alouf

Inria – Université Côte d'Azur

Working group COSMOS of the GDR RO, 22 November 2021

Outline

The prefetching game

A world of "documents": a weighted & directed graph

- nodes represent documents
- arcs represent possible transitions
- weights represent transition probabilities

Model of a random & memoryless surfer: random walk / Markov chain.

The game

- Two players: the Surfer and the Controller
- The surfer stands at a node of the graph
- Round of the game:
  - The Controller marks k nodes
  - The surfer moves to a neighbor of its position, randomly
- A cost is incurred if the surfer moves to an unmarked node.
- The game ends when all nodes are marked, or the surfer is trapped in a marked subgraph.

An optimal control problem, rather than a dynamic game:

- surfer is not strategic
- dynamics independent of control

Simplifying features

Several assumptions are made for keeping a simple model and can/could be relaxed in realistic cases:

- unlimited memory for the controller: no caching decisions
- atomic documents: no partial prefetching
- constant prefetching "budget" k:
  - document viewing time does not vary
  - bandwidth does not vary

Goal of the game

Optimization problem

Minimize the (expected) total cost where:

cost = number of times the surfer moves to an unmarked node

Feasibility problem

Given a document graph, possibly with marked nodes, and a surfer position, does there exist a marking strategy that realizes zero cost?

If the answer to the feasibility problem is "yes", it provides a solution to the optimization problem.

The feasibility problem

The feasibility problem is difficult in general: see

F. Fomin, F. Giroire, A. Jean-Marie, D. Mazauric, N. Nisse. To satisfy impatient web surfers is hard. Theoretical Computer Science, *526(20) :1–17, March 2014.*

However, it is easy in trees.

Feasibility in trees

Let a tree be represented by $t = (v_0, (t_1, \ldots, t_m))$, where v_0 is the root, m the number of sons/subtrees. Let $C(v)$ be the set of children of node v.
Consider the following recursive construction:

- $f(v) = 0$ for all leaf v
- $f(v) = \max\{0, m + \sum_{w \in C(v)} f(w) - k\}$ for all internal nodes

Feasibility in trees (ctd)

Theorem (Fomin *et al.* 2014)

There exists a zero-cost policy for tree t if and only if $f(v_0) = 0$.

Definition: f-policy

Define the policy from function $f(\cdot)$: v being the position of the surfer,

- mark the sons, if not already marked
- mark $f(w)$ unmarked nodes in the subtree with root w, for all $w \in C(v)$, in a connected way

Proof of Theorem

$f(v_0) = 0$ iff the f-policy realizes zero cost.

Open problem

Optimality of the zero-cost policy?

Is the f-policy optimal for the Optimization Problem when it fails to solve the Feasibility Problem?

Progress

Dynamic graphs

It is relevant to consider that the document graph is dynamic

- it may be too large to be stored: *e.g.* the web graph
- the optimal decision involves the whole graph: too much information for the decision
  - purposedly forget nodes that are "far away" from the surfer

Dynamic trees

The simplest dynamic graph with given "lookahead": random trees of depth d.
When the surfer moves to a node v:

- nodes at distance $d+1$ are removed
- leaves at distance $d-1$ are completed with a random number of new leaves

The process becomes endless $\rightarrow$ finite-horizon or stationary-like criterion.

A first exploration

In his 2019 study, Q. Petitjean has defined and compared strategies based on indices.

- f-based index: compute recursively

$$f(v) = \max\{0, \bar{\mu}(C(v)) + \sum_{w \in C(v)} f(w) - k\}$$

  where $\bar{\mu}(A)$ is the number of unmarked nodes in set A.
- second-generation index: $s(v) = |C(v)|$
- middle-term index: compute recursively

$$ind(v) = \frac{1}{100 \times |C(v)|} \left(\bar{\mu}(C(v)) + \sum_{w \in C(v)} ind(w) \right)$$

A first exploration (end)

Algorithm:

- mark sons with higher indices first
- if budget remains, place a mark in the subtree with higher index, recursively
- if budget still remains, recompute and redo.

Conclusion:

- "It is observed that no strategy is really more efficient than the others"
- The short-term strategy is slighty more efficient for prefetching budgets larger than $p/2$

This calls for a deeper study...

Modeling the decision problem as a MDP

Construction of a MDP with decision/transition cycle:

1. the surfer stands at the root of a depth-d tree
2. the controller marks up to k nodes
3. the surfer moves randomly and uniformly to one of the depth-1 nodes
4. the subtree is completed by converting each leaf into a depth-1 tree with random number of leaves, uniform between 1 and p.
5. the rest of the tree is forgotten (including former root)

Note: the surfer never goes back.

MDP specifications

The MDP has:

- State space: set of marked trees of depth d and arity p, with unmarked leaves
- Action space: marking at most k nodes
- Transitions: deterministic concerning the controller, random for the surfer and the tree

$$
\begin{aligned}
t &= (\mu, (s_1, s_2, \ldots, s_m)) \\
&\to (\mu, (s'_1, s'_2, \ldots, s'_m)) && \text{after marking} \\
&\to s'_r = (\mu_r, (\sigma_1, \ldots, \sigma_\ell)) && \text{with probability } 1/m \\
&\to t' = (\mu_r, (\sigma'_1, \ldots, \sigma'_\ell)) && \text{with probability } (1/p)^{\text{leaves}(s'_r)}
\end{aligned}
$$

- Cost: 1 if $\mu_r = 1$, 0 otherwise

Criterion and method of proof

We chose as criterion the expected, infinite horizon average cost. The dynamic optimality equation (DOE, aka "Bellman") and its relation to optimal policies are:

Theorem (Ross)

If there exists a bounded function $f(s)$ for every $s \in S$ and a constant g such that

$$g + f(s) = \min_a \left[c(s,a) + \sum_{s' \in S} P(s,a,s') f(s') \right]$$

then there exists a stationary policy γ^* such that, for all s,

$$g = \max_\gamma \phi_\gamma(s) = \phi_{\gamma^*}(s).$$

First results

Definition

A policy that marks unmarked sons in priority is called "greedy".

Theorem

If $k = 1$ and $d = 2$, or if $d = 1$, any greedy policy is optimal.

Both results can be proved for the finite-horizon, then pass to the limit.

How to prove it using the DOE?

- find g
- find f
- check DOE

Cost of greedy policies

To evaluate g, a detour via Markov chains is useful.
A tree shape is a tree stripped of its marks (or unmarked).

Property

The process of tree shapes is independent of the control. It is a Markov chain with stationary distribution that of Galton-Watson trees stopped at depth d.

Cost of greedy policies (ctd)

Corollary

Let t be distributed according to this stationary distribution, and let C be the random variable:

$$C = \frac{[\,|s(t)| - k\,]^+}{|s(t)|}.$$

Then,

$$\mathbb{E}C \;=\; \frac{1}{p}\,\mathrm{H}_{pk}, \qquad k \le p,$$

where

$$\mathrm{H}_{pk} \;:=\; \sum_{m=k+1}^{p} \frac{m-k}{m} \;=\; p - k - k(\mathrm{H}_p - \mathrm{H}_k).$$

Back to optimality

The value of g is given by $\mathbb{E}C$ thus computed.

The value of f is: $f(\mu, \underbrace{(1, \ldots, 1)}_{j \text{ times}}) = \frac{(j-k)^+}{j}$, when $d = 1$

and when $d = 2$, $k = 1$:

$$f(\mu; (\mu_1, j_1), \ldots, (\mu_m, j_m))$$
$$= \begin{cases} -\frac{1}{m}\left(1 + \sum_{r=1}^{m} \mu_r + \sum_{r=1}^{m} \frac{1}{j_r}\right) & \sum_{r=1}^{m} \mu_r \leq m-1 \\ -1 - \frac{1}{m}\left(2\sum_{r=1}^{m} \frac{1}{j_r} + |\{r | j_r = 1\}| \frac{\mathrm{H}_p - p}{p-1}\right) & \sum_{r=1}^{m} \mu_r = m. \end{cases}$$

Proving optimality

Annoying feature of this method of proof:

- The function f has to be guessed for states that "don't really matter": they don't belong to the recurrent class of Markov chains derived from optimal policies.
  Specifically: these states have marks where an optimal policy would never place one.
- This makes the extension to the case $k = 1$, $d > 2$ cumbersome, although the result looks simple.

Findings

The case $d = 2$ and $k = 2$.
Contrary to the "easy" cases just solved, there is now the possibility to mark depth-2 nodes in optimal policies:

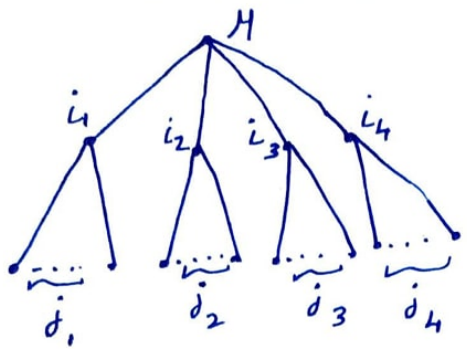
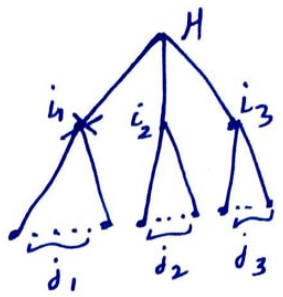
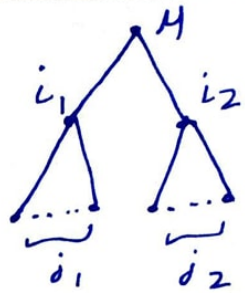
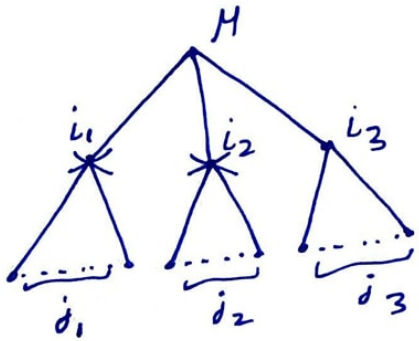
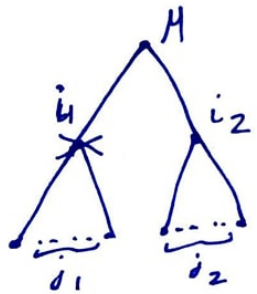
- either because the root of the tree has only one son
- or because some of the depth-1 nodes are already marked

Exploration of greedy policies

All policies mark first as many unmarked sons as possible.

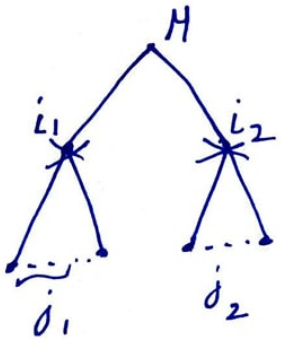
- **Greedy Depth 1**: Only the sons of the tree are marked.
- **Greedy Smallest**: If budget remains, then mark the first leaf of the smallest subtree. If budget still remains, mark the second leaf of the smallest subtree, if any. Otherwise, mark the first leaf of the second smallest subtree.
- **Greedy Largest**: If budget remains, then mark the first leaf of the largest subtree. If budget still remains, mark the second leaf of the largest subtree, if any. Otherwise, mark the first leaf of the second largest subtree.
- **Greedy Leftmost**: If budget remains, then mark the first leaf of the leftmost subtree. If budget still remains, mark the second leaf of the leftmost subtree, if any. Otherwise, mark the first leaf of the second leftmost subtree.

Typology for depth-2 trees

TREE TYPE LABEL	EXAMPLES
Type 1: $m \geq \sum_{q=1}^{m} i_q + 2$	 $1 \leq d_1, \dots, d_4 \leq p$  $i_1 = 1$ $1 \leq d_1, d_2, d_3 \leq p$  $1 \leq d_1, d_2 \leq p$
Type 2: $m < \sum_{q=1}^{m} i_q + 2$ **SUBTYPES:** **2a:** Trees with $(m, \sum_{q=1}^{m} i_q) = (3,2)$ $(2,1)$ $(1,0)$	 $i_1 = i_2 = 1$ $1 \leq d_1, d_2, d_3 \leq p$  $i_1 = 1$ $1 \leq d_1, d_2 \leq p$  $1 \leq d_1 \leq p$

Typology for depth-2 trees

2b: $(m, \sum_{s=1}^{m} i_s) = (2,2)$



$i_1 = i_2 = 1$

$1 \leq d_1, d_2 \leq b$

2c: $(m, \sum_{s=1}^{m} i_s) = (1,1)$

$i_1 = 1$

$1 \leq d_1 \leq b$

The Greedy Finite Optimal policy

Tree Type	Unmarked Sons	Spec. subtree sizes	Optimal Action
Type 1	≥ 2		$a(d1, d1)$
Type 2a	1	$j_r \geq 3$ for some r	$a(d1, l_{j_c})$, $j_c = \min_r j_r \geq 3$
		$j_r < 3$ for all r	$a(d1, l_{j_c})$, $j_c = \max_r j_r$
Type 2b	0	$j_1 > 3$	$a(l_{j_1}, l_{j_1})$
		$j_1 \leq 3$	$a(l_{j_1}, l_{j_2})$
Type 2c	0		$a(l_{j_1}, l_{j_1})$

Specification of the Greedy Finite Optimal policy. Conventions:

- $j_2 \geq j_1$
- d1 stands for "depth 1"
- l_j stands for "leaf of subtree of size j"

Results for greedy policies

Average cost of the different greedy policies

Policy	$p = 3$	$p = 4$	$p = 5$
Greedy Depth 1	0.111111	0.208333	0.286667
Greedy Smallest	0.067912	0.161568	0.229741
Greedy Leftmost	0.062802	0.160227	0.226289
Greedy Largest	0.054369	0.156907	0.217443
Greedy Finite Optimal	0.054369	0.154401	0.208282

Optimal policy for $d = 2$ and $k = 2$

Theorem (?)

The Greedy Finite Optimal policy is optimal for $d = 2$ and $k = 2$, when $p = 2, 3, 4, 5$.

Proof: numerically check that this policy γ is a fixed point of the Policy Iteration algorithm (Puterman, chapter 8.6).
Equivalently, find g and f solving:

$$0 = c_\gamma + g1 + (P_\gamma - I)f$$

and check that those solve the DOE.
Possible weakness: done only for "relevant" states.

Conclusions and issues

Conclusions:

- Greedy (sons-first-marking) policies are possibly optimal
- Connected policies are possibly optimal
- Next steps planned:
  - Find direct arguments for the optimality of greedy policy (including $k = 1$!)
  - Check numerically the optimality of "Greedy Finite Optimal" policy for larger p
  - Use agregation to "factorize" the MDP