Chunking Tasks for Present-Biased Agents

Joe Halpern and Aditya Saraf

Cornell University

EC'23
December 30, 2023

Present Bias: A Shipping Story

You need to pack and ship an item you just sold to the buyer..

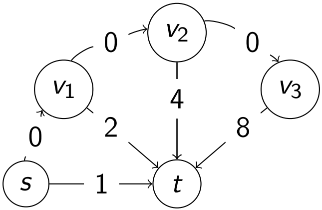
Present Bias: A Shipping Story

You need to pack and ship an item you just sold to the buyer..
but you'll do it tomorrow.

(Naive) Present Bias in Task Completion

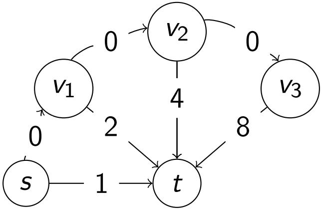
Model tasks as a directed, acyclic graph with start node s and end node t [Kleinberg and Oren, 2014].

Shipping example:



(Naive) Present Bias in Task Completion

- (Biased) agents reason *locally*: they multiply cost of next edge by b
- (Biased) agents reason *naively*: they assume they will behave optimally in the future



- Biased agents can take exponentially more expensive paths than optimal agents, in the worst case

Related Work

Abandonment model:

- task designers can delete parts of the graph $\implies$ NP-hard to find *motivating subgraphs* [Tang et al., 2017]
- task designers can place rewards on intermediate vertices $\implies$ NP-hard to allocate rewards [Albers and Kraft, 2019, Tang et al., 2017]

[Saraf et al., 2020] uses competition to lower the cost incurred by biased agents.

Present Bias: Chunking for Shipping

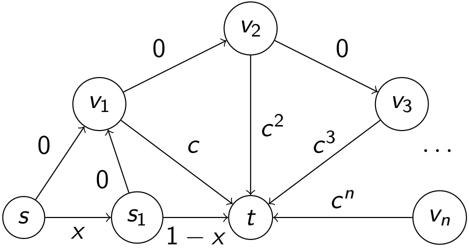
With chunking:
You need to pack and ship an item you just sold to the buyer. Rather than thinking of this as a large, single task, you conceptualize the packing and shipping tasks separately. Now, the packing doesn't seem so daunting on its own, and shipping a packed item is also less daunting.

Another motivating application:

- You're an instructor, and tell students how to break a complicated assignment into more manageable pieces.

Chunking Model

- Select edges to chunk
- For each edge (u, v) to chunk:
- Break (u, v) into multiple pieces, and set the cost of each chunk (goal: agent prefers the chunked edge)
- New chunking vertices add all edges $(u, w)_{w \neq v}$ in the original graph



- Adding all of u's connections models the fact that the chunking is not strictly enforced, you can back out of it

Simplifying Assumptions

- Continuous chunking of edges is an approximation for the more realistic discrete setting
- Chunking has no overhead cost to agents
- Known bias
- No abandonment possible

Edges Along the Shortest Path

- "Optimal" edge-chunking: one that minimizes the agent's perceived cost to t starting with that chunking
  - equiv. one that persuades an agent of maximal bias to take the chunking

Theorem

Suppose we partition a shortest-path edge of cost x into k chunks. Let x_i be the cost of the ith chunk. The optimal chunking sets:

$$x_i = \frac{(b-1)^{k-i} b^{i-1}}{b^k - (b-1)^k} x.$$

- Earlier chunks are easier; chunks become progressively more difficult
- Example: let $b = 2, k = 4$. The four chunks should cost $1/15$, $2/15$, $4/15$, and $8/15$ of the total weight (in order)
- Balances the agent's perceived cost to t starting at each chunk

Non short-path edges

- *Short-path edge*: an edge (u, v) such that the shortest path from u to t takes edge (u, v)
- Naive agents assume that they will take the shortest path from the next vertex, so for short-path edges, their perceived cost to t starting at earlier chunks is a function of the cost of later chunks
  - Not true for non-short path edges
- Why chunk non-short path edges?
  - Sometimes necessary to reduce the agent's cost

Non short-path edges: high-level solution

- Basic idea: start by evenly splitting the cost among all chunks
- Complication: later chunks might actually become part of the shortest path (and so an even split would not be optimal)
- Solution: split the cost for such later chunks according to Theorem 1 (increasing costs)
- We provide an algorithm to solve this problem

Local Budget

- Budget: can spend up to k chunks on each edge
- Goal: chunk the graph so that the agent takes the path with lowest (real) cost possible
- Naïve solution: chunk all edges with k chunks with the optimal algorithm.
  - Problem: shouldn't chunk edges that the agent should avoid
- Instead, should find the cheapest path that it's possible to persuade the agent to take, and just chunk that
- What's relevant? Whether we can persuade the agent to take an edge or not.
- Variation of shortest path. $\mathcal{P}(u)$ is the set of neighbors of u that we can convince the agent to go to with k chunks.

$$cost[u] = \min_{v:(u,v)\in\mathcal{P}(u)} c(u,v) + cost[v]$$

Global Budget

- Can spend up to k chunks on the whole graph
- What's relevant? For all (u, v), compute $l_{u,v}$, the minimum number of chunks needed to persuade the agent to take (u, v) (∞ if impossible)
- Very similar recurrence to before.

$$cost[u, i] = \min_{v:(u,v)\in E, l_{u,v}\leq i} c(u, v) + cost[v, i - l_{u,v}]$$

Cost Ratio

Cost ratio: ratio of the biased agent's cost over the optimal cost.

- Exponential (b^n) without chunking

Given an agent with bias b, and a local chunking budget of k, what is the highest cost ratio over all optimally-chunked graphs?

Theorem

Define $b_{\min}$ as $\frac{1}{1-\left(\frac{b-1}{b}\right)^k}$. If G' is an optimal chunking of G with local budget k, then the cost ratio for agent in G' is at most $b_{\min}^n$.

- Effective bias: b-agent behaves in chunked graph as $b_{\min}$-agent behaves in original graph.

Corollary

Given a local budget of $k = O(n)$, the optimal chunking G' of G has a constant cost ratio.

Two Agents

- You're an instructor. Some of your students procrastinate a lot, some procrastinate only mildly. You need to present your class with a single chunking that works well for all of them.
- Goal: minimize the sum of the agents' cost with a single chunking of the task graph. Agents only differ in their bias parameters
- Complication: when the agents are at the same vertex, we need to figure out how to (1) keep agents together and (2) split agents up
- (2) is difficult. Intuition: chunking for one agent makes the edge more appealing for the other agent
- We provide an algorithm that solves optimal two-agent graph chunking, but does not generalize well to more agent types.

Conclusion

Main results:

- To chunk a single edge, make earlier chunks easier and later chunks more difficult
  - Provides some formal justification for the common folk wisdom to “start off easy”
- With a linear number of chunks on each edge, we can reduce the cost ratio from an exponential factor to a constant factor
- Chunking for even two agents becomes much more complicated

Future directions:

- Chunking for more than two agents
- “Checkpoints” instead of chunking