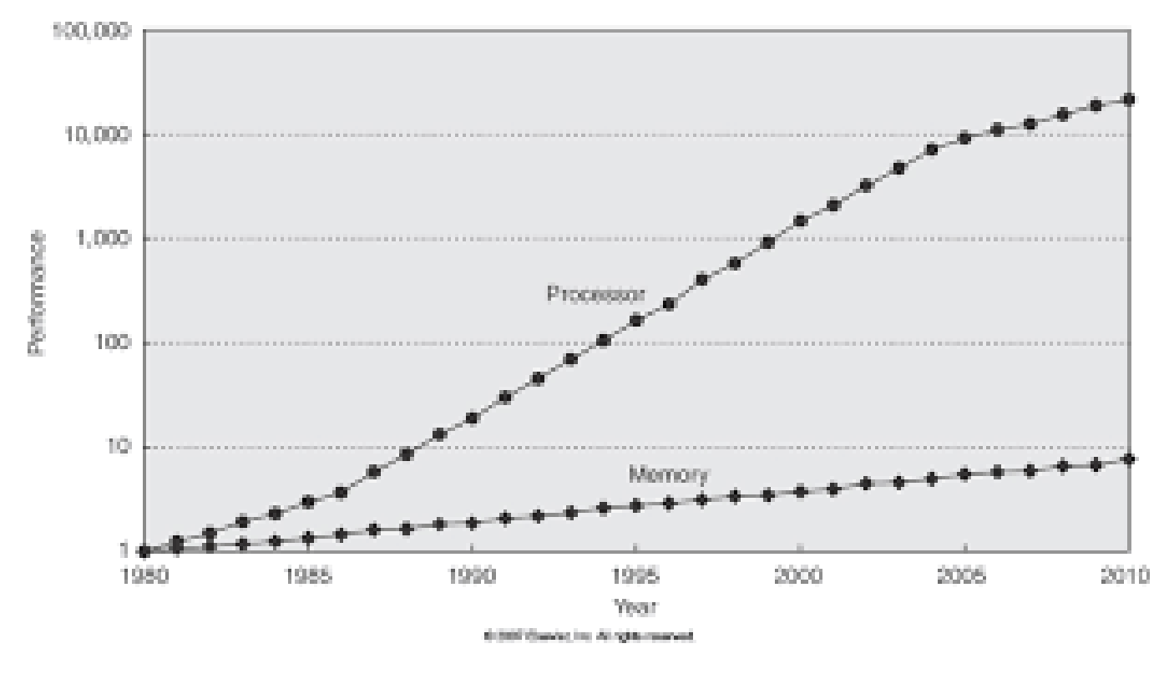
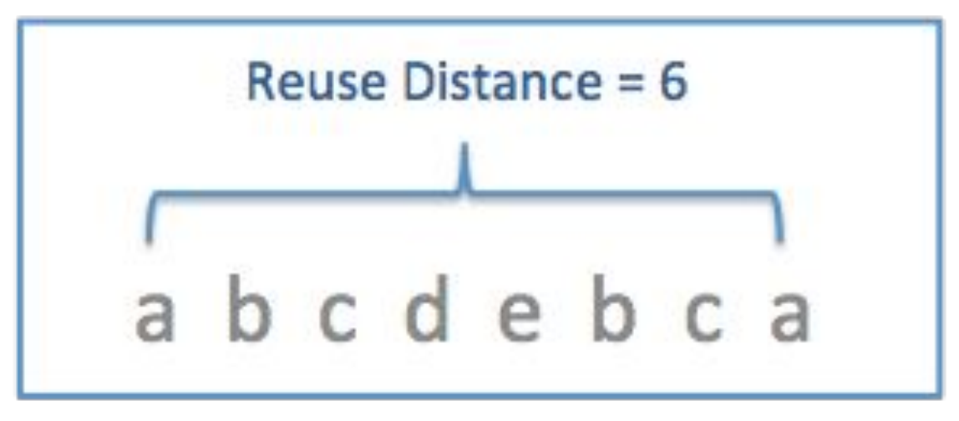


Reuse-based Adaptive Caching

Isaac Grosof and Katherine Cordwell
15-740

The RDD and Caching

For a given cache line ℓ, its **reuse distance** is the number of accesses to a cache set between two consecutive accesses to ℓ. For an example, see the figure below.



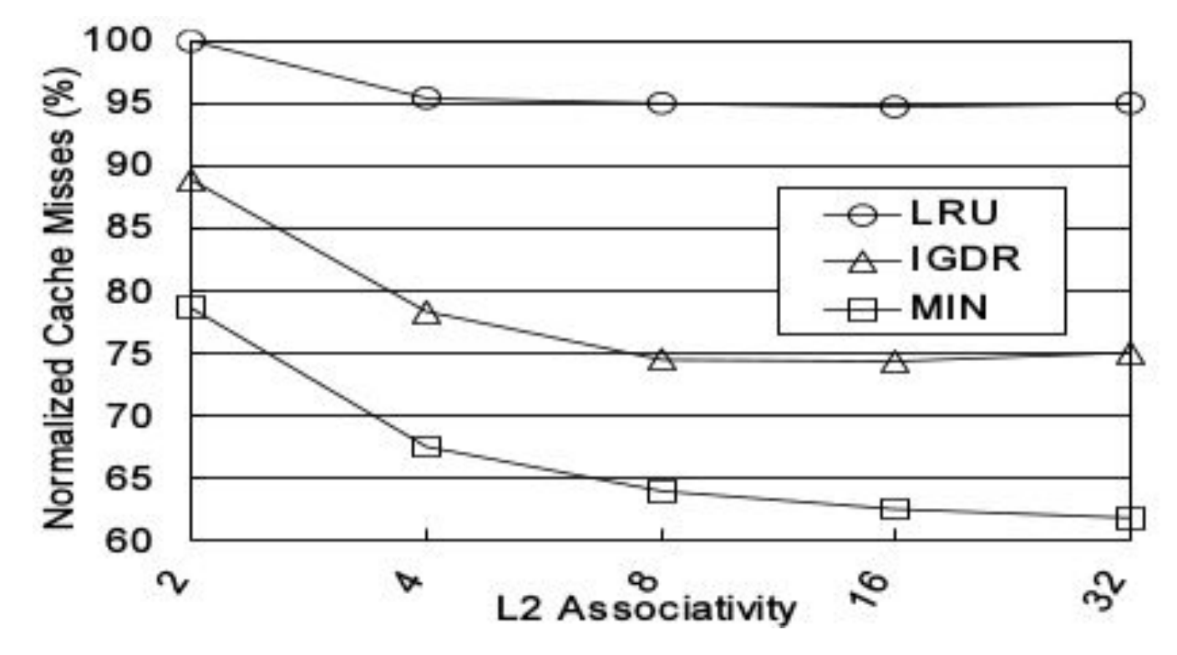
For a given cache line, we can also consider its distribution over possible reuse distances. This distribution is called the **RDD**. The RDD gives us the probability associated with each potential future reuse distance.

Many caching policies are based on an approximation of the RDD. Approximation methods are highly nonuniform across policies.

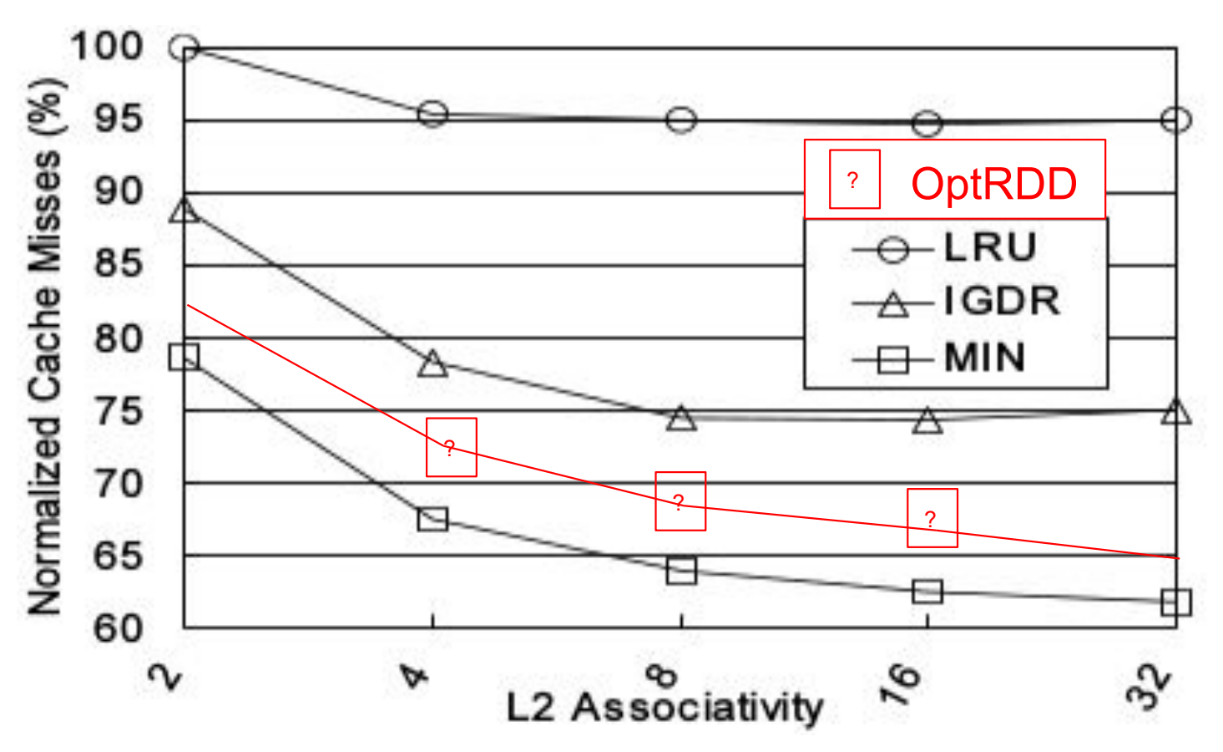
Our Question

Current policies often compare themselves to Belady's MIN, which is the offline optimal policy that knows the future. However, Belady may not be the right baseline comparison for RDD-based policies. In particular, it gives no information about how much information was lost due to approximation error.

Accordingly, we are interested in the following question: Given access to the full RDD, what is the best caching policy? In other words, instead of a comparison like this (which we've taken from Takagi and Hiraki's paper on Inter-reference Gap Distribution cache replacement),



we want to have a comparison like this:



as this gives an indication of how good IGDR could have been, given that it is an RDD-based policy.

The Gittins Index

Towards this goal, we develop a Gittins index policy. In this policy, each line has a hit rank, which is defined as its hit chance over its occupancy time. At any step, we evict the line with the lowest rank.

As a math equation, this translates to

$$\min_{\ell} \max_{b \in \operatorname{supp} D - age_\ell} \frac{\Pr(D - age_\ell \le b \mid D > age_\ell)}{E(\min(D - age_\ell, b) \mid D > age_\ell}$$

In a situation where we can evict lines whenever we want, Gittins is optimal.

Simulation Parameters

We make the following assumptions:

- Fully associative, inclusive cache
- Every line has a discrete RDD
- There is a fixed number of discrete RDDs (and some probability distribution over those RDDs)
- A line is inserted into cache with a lifetime sampled from its RDD

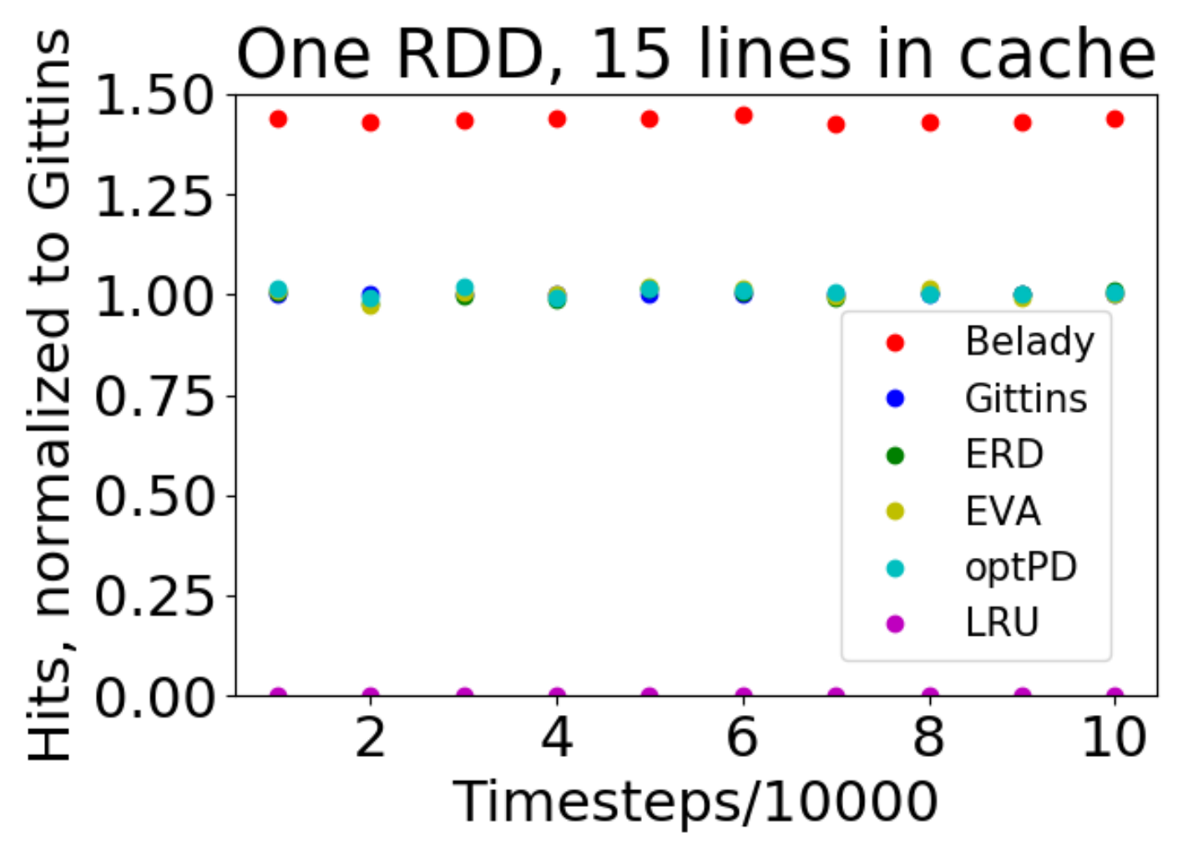
We compare the following policies:

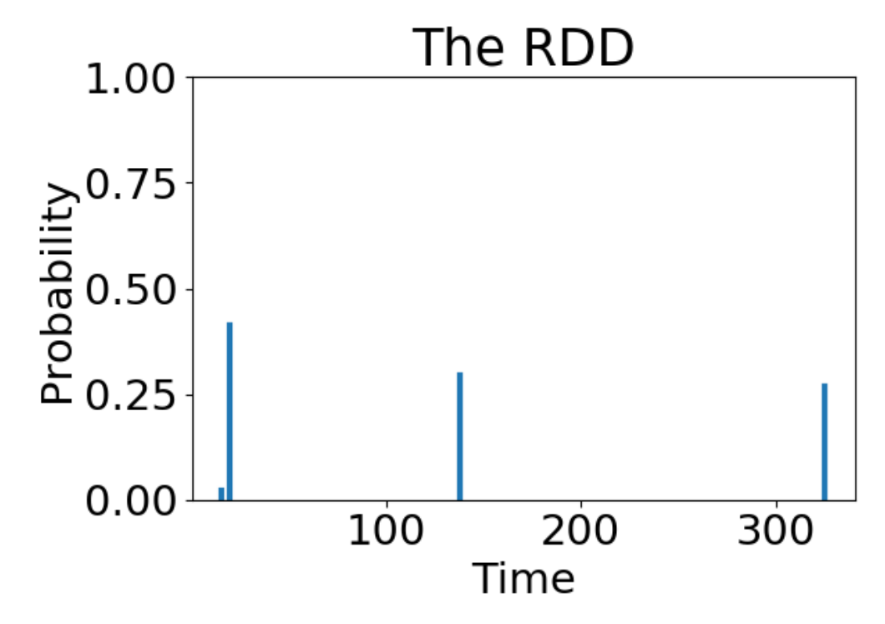
- Gittins
- Belady's MIN
- LRU
- Expected Reuse Distance (ERD)
- BRRIP/SRRIP-FA, SRRIP-HP
- Economic Value Added (EVA)
- Optimal static protection distance (optPD)

In optPD, we run static PDP with all feasible protection distances, and pick the best one.

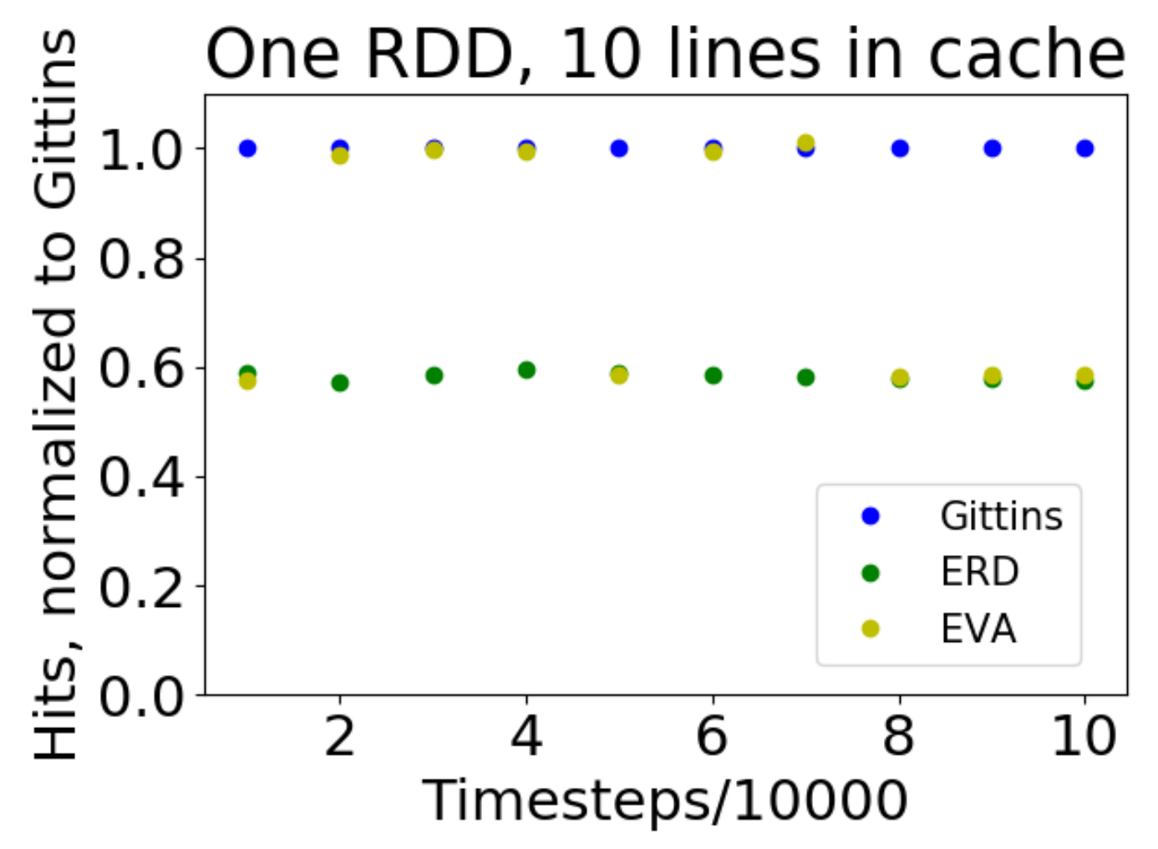
Results (One RDD)

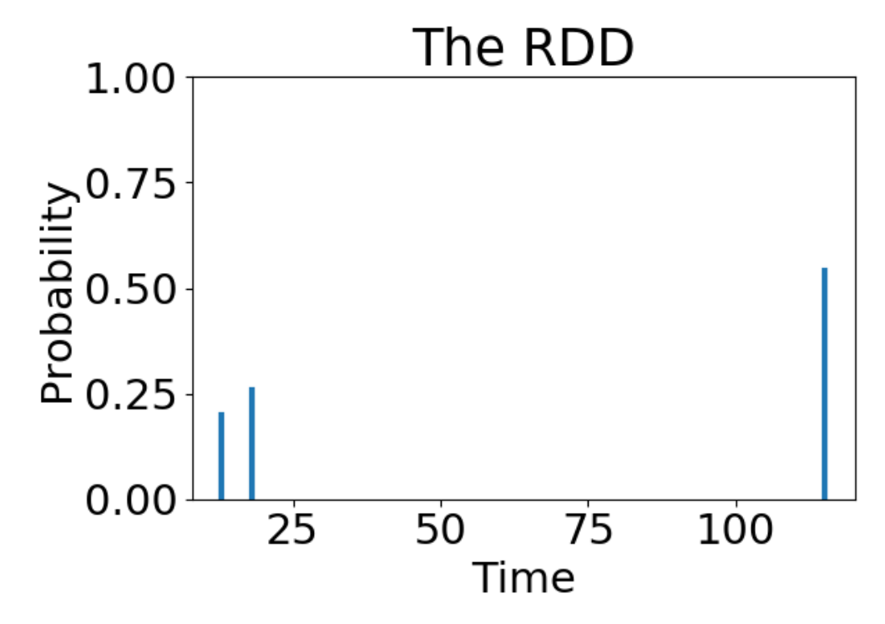
Here is some data after running experiments with a single, "randomly generated" RDD, with hit counts normalized to the hit count of Gittins. It is relatively representative of general behavior, although LRU can do better depending on the cache size.



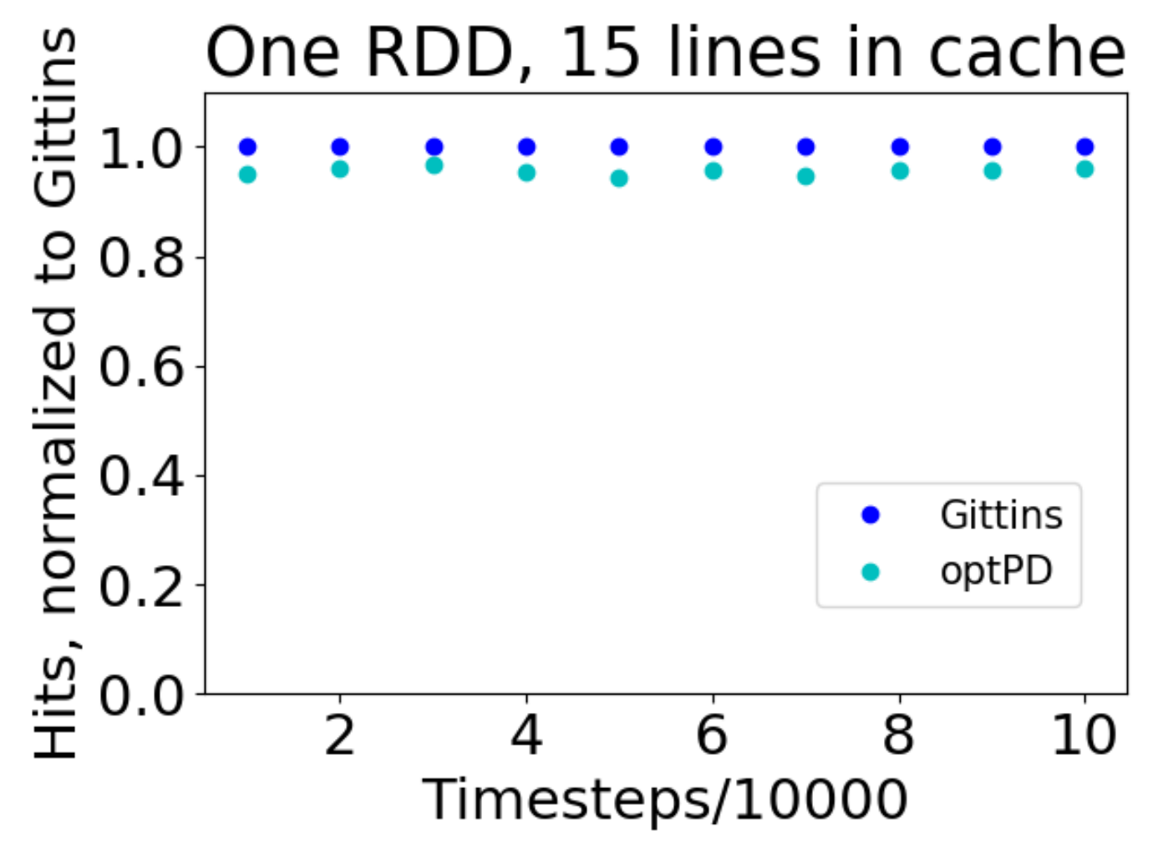


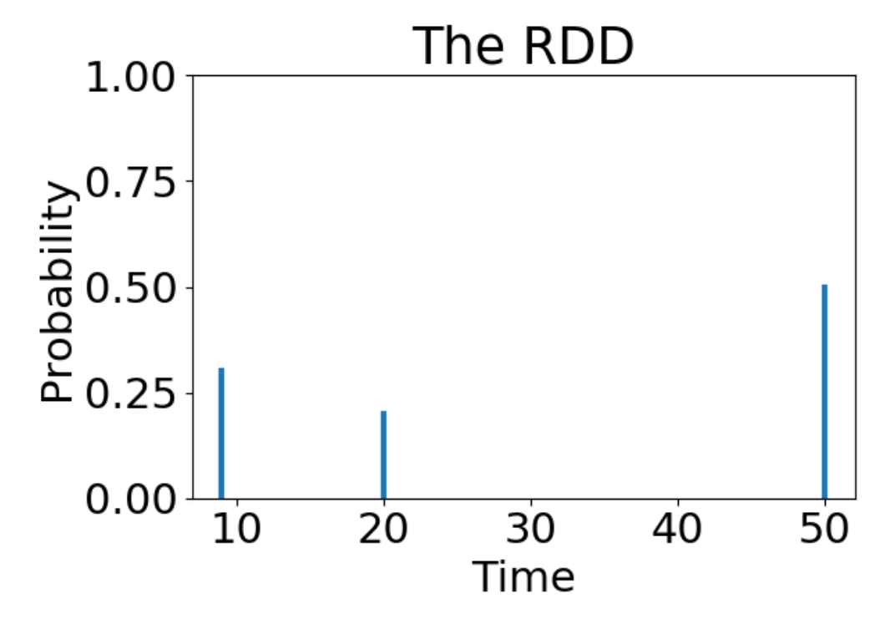
Also, there are some examples where ERD and EVA perform poorly. One of these is shown below. Here, it seems that EVA does not always explore the future enough to know that things will hit at time 18. We can fix this by seeding EVA with the choices that Gittins would have made for a small percentage of timesteps.





Additionally, there are rare examples where optPD performs poorly. One is shown below.

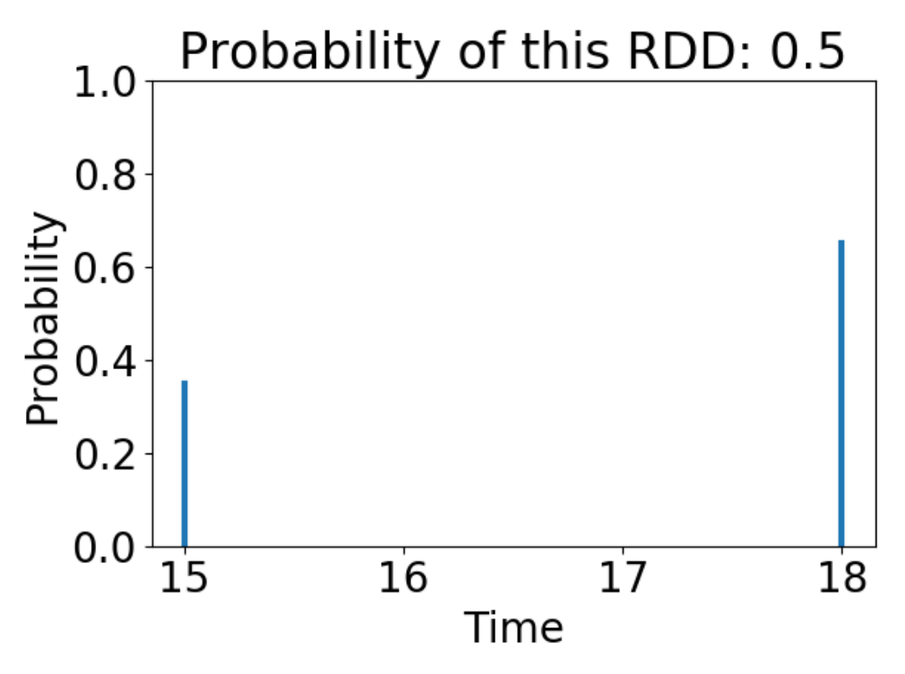


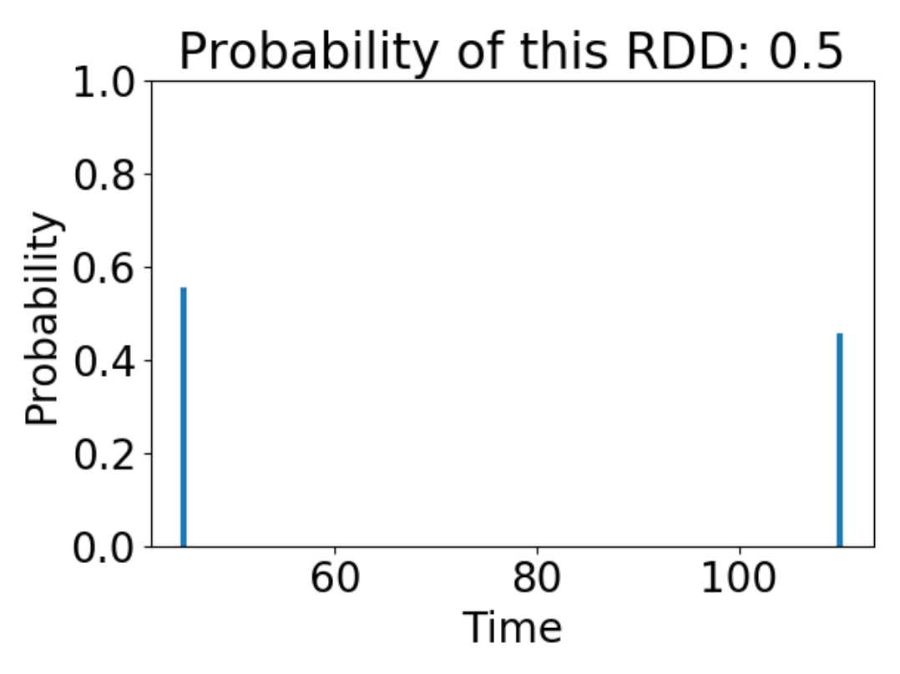


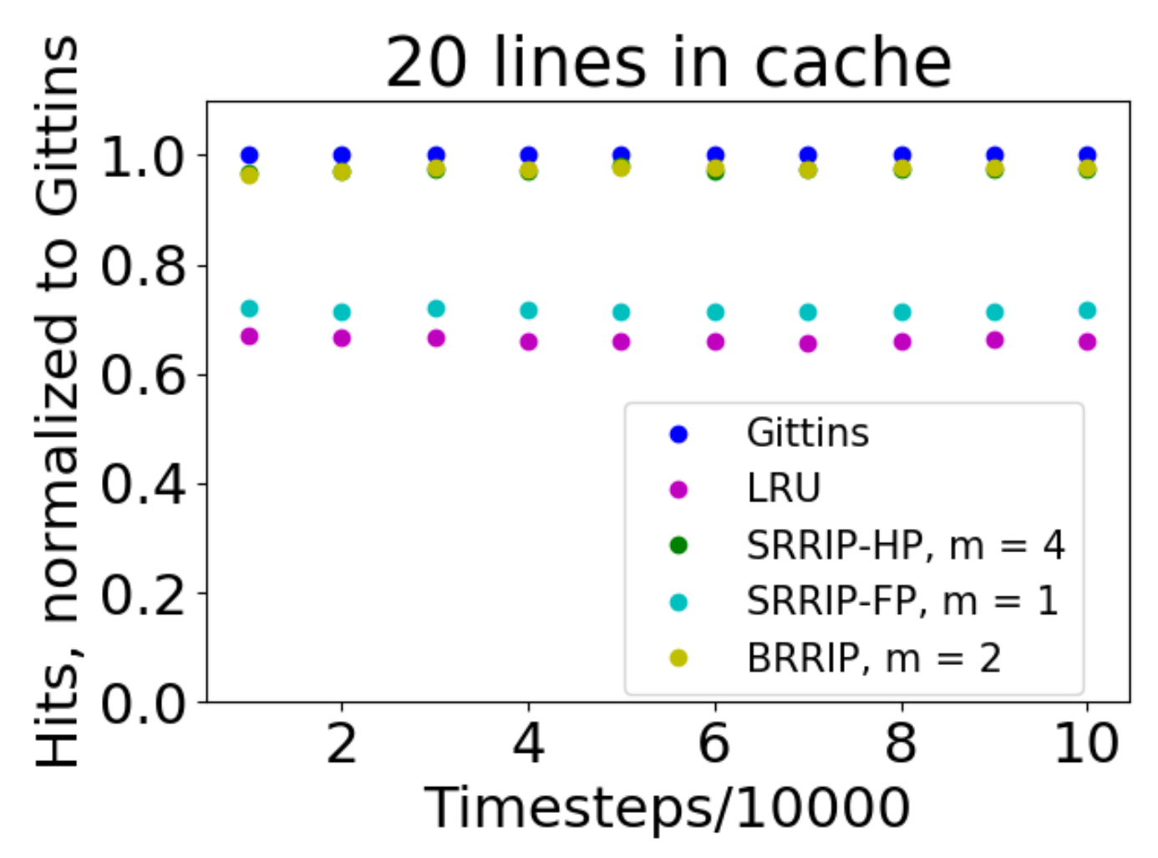
Results (Multiple RDDs)

Some policies only really make sense to evaluate in the case where we have multiple RDDs.

For example, with a single RDD, it doesn't make sense to compare LRU, SRRIP, or BRRIP to Gittins and other RDD-based policies, because in this case LRU, SRRIP, and BRRIP don't make any use of the RDD and have no chance of performing well. With multiple RDDs, and given a sufficient number of cache lines, it is slightly more fair to compare LRU, SRRIP, and BRRIP to Gittins, because by design, these policies will "learn" some information about the RDDs, in the sense that they are more likely to keep lines with RDDs that are likely to get many hits than lines with RDDs that are likely to get few hits. We can see this in the test case below.







However, it no longer makes sense to compare optPD to Gittins when there are multiple RDDs, because optPD is essentially designed to be a good policy when there is one RDD, and its performance is not only computationally infeasible but also significantly degraded in the case where there are many disparate RDDs.

Takeaways

We can get more nuanced comparisons if we classify policies according to what information they are using and evaluate them against policies that use similar information.

For example, Belady's MIN is an unrealistic comparison point for RDD-based policies, but it would be a fair comparison point for policies that are based on the program counter.

With a single RDD, Gittins, optPD, EVA, and ERD are comparable, but it is not necessarily fair to compare Gittins to SRRIP, BRRIP, or LRU. With multiple RDDs, Gittins, SRRIP, BRRIP, LRU, EVA, and ERD are comparable, but it is not necessarily fair to compare Gittins to optPD.

Gittins seems to be the best RDD-only policy.

What Next?

Some things we may consider next are:

- More multiple RDD experiments.
- Modifying Gittins to take into account that in a cache, lines may linger beyond when we expect them to be evicted.