

Replication Strategies in Unstructured Peer-to-Peer Networks

Edith Cohen

Scott Shenker

Some slides are taken from the authors' original presentation

Search strategies in P2P Networks

Search is performed by probing peers

- **Structured (DHTs):** (Freenet, Chord, etc) location is coupled with topology - search is routed by the query. Only exact-match queries, tightly controlled overlay.
- **Unstructured:** (Gnutella, FastTrack); search is “blind” - probed peers are unrelated to query. Resilient to transient peers; versatile queries

Replication in P2P architectures

- No proactive replication (Gnutella)
 - Hosts store and serve only what they requested
 - A copy can be found only by probing a host with a copy
- Proactive replication of “keys” (= meta data + pointer) for search efficiency (FastTrack, some DHTs)
- Proactive replication of “copies” – for search and download efficiency, anonymity. (Freenet)

The Focus of this paper

How to use replication to **improve search efficiency** in unstructured networks with a **proactive replication** mechanism ?

Search and replication model

Unstructured networks with replication of keys or copies. Peers probed are **unrelated** to query/item. Success likelihood can not be better, on average, than random probes.

- **Search**: probe hosts, uniformly at random, until the query is satisfied (or the search max size is exceeded)
- **Replication**: Each host can store up to r copies (or keys = metadata + pointer) of items.

Goal: minimize average *search size* (number of probes till query is satisfied)

Search size

Query is **soluble** if there are sufficiently many copies of the item. Query is **insoluble** if item is rare or non-existent.

What is the **search size** of a query ?

Insoluble queries: maximum search size

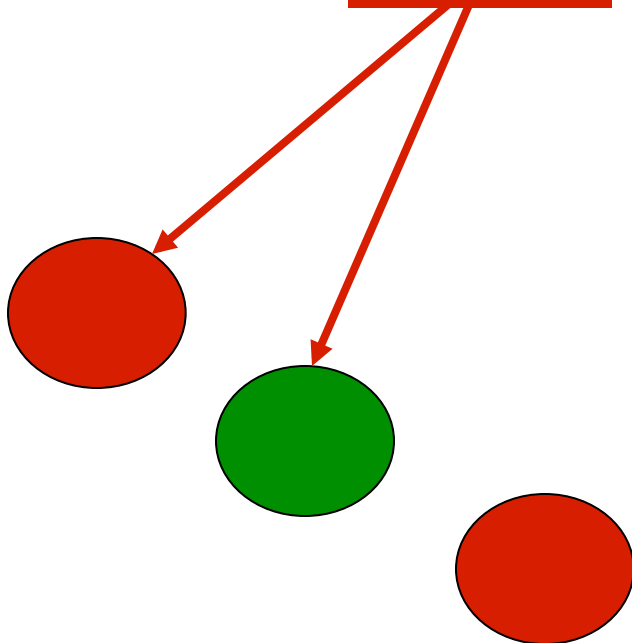
Soluble queries: number of visited nodes until answer is found.

We look at the **Expected Search Size** (ESS) of each item. The ESS is inversely proportional to the **replication factor** (fraction of peers with a copy of the item).

Search Example

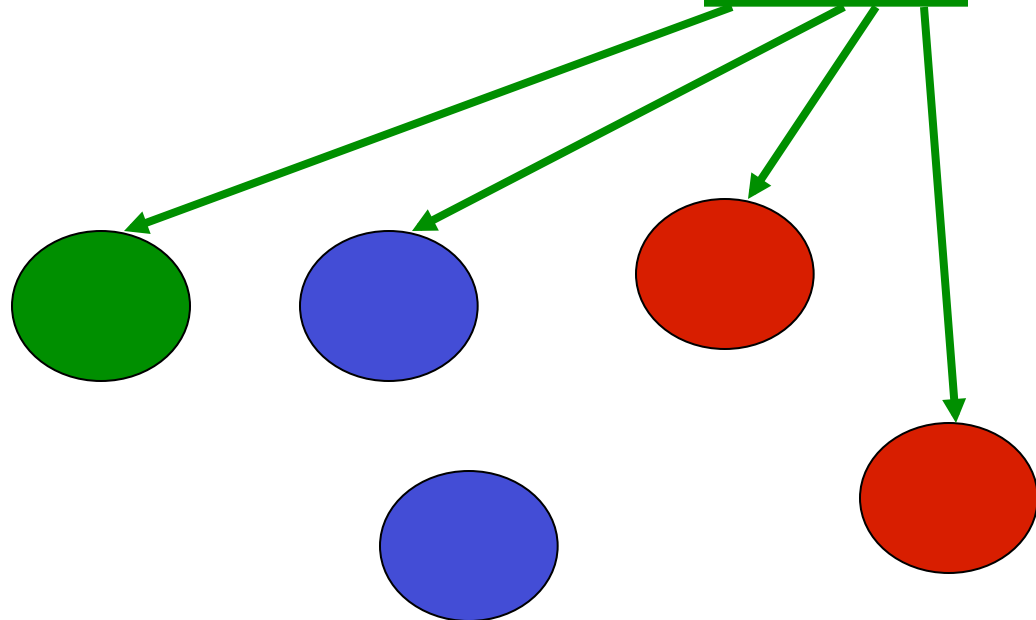
Looking for the green object

2 probes



Random walk is sometimes a popular tool

4 probes



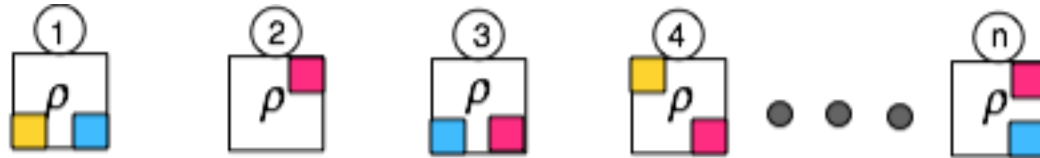
Expected Search Size (ESS)

Consider m items with relative *query rates*

$$q_1 > q_2 > q_3 > \dots > q_m. \quad \sum_i q_i = 1$$

- *Allocation* : $p_1, p_2, p_3, \dots, p_m$ $\sum_i p_i = 1$
 i^{th} item is allocated p_i fraction of storage. (keys placed in p_i fraction of hosts)
- Search size for i^{th} item is a geometric random variable with mean $A_i = 1/(\rho p_i)$.
- ESS is $\sum_i q_i A_i = (\sum_i q_i / p_i) / \rho$

Model and notations



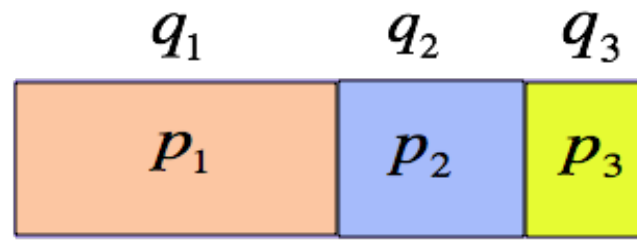

m objects

Object j has r_j copies

$$p_i = \frac{r_i}{R} = \frac{r_i}{n\rho} = \frac{\text{Population density of object } i}{\rho}$$

Query vector $q = (q_1, q_2, q_3, \dots, q_m)$

Model and notations



Total space = 1

Allocation vector $p = (p_1, p_2, p_3, \dots, p_m)$

Query vector $q = (q_1, q_2, q_3, \dots, q_m)$

q_i = normalized query rate for object i

$$q_1 \geq q_2 \geq q_3 \geq \dots \geq q_m$$

$$\sum_{j=1}^{j=m} p_j = 1 \quad \sum_{j=1}^{j=m} q_j = 1$$

What is an allocation?

Allocation means **how many copies** of each object will be made. A natural question is to **relate this to the query rate**. So

Allocation: $q \rightarrow p$

Uniform and Proportional Replication

Uniform Allocation: $p_i = 1/m$

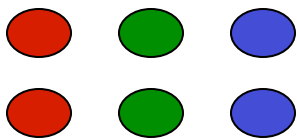
- Simple, resources are divided equally

• **Proportional Allocation:** $p_i = q_i$

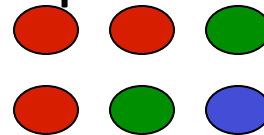
- “Fair”, resources per item proportional to demand
- Reflects current P2P practices

Example: 3 items, $q_1=1/2$, $q_2=1/3$, $q_3=1/6$

Uniform



Proportional



Basic Questions

- How do Uniform and Proportional allocations perform/compare ?
- Which strategy minimizes the **Expected Search Size** (ESS) ?
- Is there a simple protocol that achieves **optimal replication** in unstructured P2P networks ?

Insoluble queries

- Search always extends to the maximum limit size.
- If we fix the available storage for copies, the query rate distribution, and the number of items that we wish to be “locatable”, then
- The maximum required search size depends on the smallest allocation of an item. Thus,
- This means that uniform allocation minimizes this maximum and thus the cost induced by insoluble queries.

Soluble queries

What about the cost of soluble queries?

Answer is more surprising ...

Soluble queries

ESS of an object is derived from a geometric distribution.

ESS of an object is inversely proportional to population density of that object.

Thus $ESS \propto 1/\rho$

ESS under Uniform and Proportional Allocations (soluble queries)

- **Lemma:** The ESS under either Uniform or Proportional allocations is m/ρ
 - Independent of query rates (!!!)
 - Same ESS for Proportional and Uniform (!!!)
- **Proof...**

Proportional:

$$\text{ESS is } (\sum_i q_i / p_i) / \rho = (\sum_i q_i / q_i) / \rho = m / \rho$$

Uniform:

$$\text{ESS is } (\sum_i q_i / p_i) / \rho = (\sum_i m q_i) / \rho = (m / \rho) \sum_i q_i = m / \rho$$

Space of Possible Allocations

Proportional Allocation means $q_{i+1}/q_i = p_{i+1}/p_i$ (“fair share”)

Uniform Allocation means $p_{i+1}/p_i = 1$

In-between allocation is: $(q_{i+1}/q_i < p_{i+1}/p_i)$ and $(p_{i+1}/p_i < 1)$
(less popular gets **more than** its “fair share”)

Other possibilities: $p_{i+1}/p_i > 1$ (more popular gets less) OR
 $q_{i+1}/q_i > p_{i+1}/p_i$ (less popular gets less than “fair share”)

Space of Possible Allocations

Theorem1: All (strictly) in-between strategies are (strictly) better than Uniform and Proportional

Theorem2: p is worse than Uniform/Proportional if
for all i , $p_{i+1}/p_i > 1$ (more popular gets less) OR
for all i , $q_{i+1}/q_i > p_{i+1}/p_i$ (less popular gets less than “fair share”)

Proportional and Uniform are the **worst**
“reasonable” strategies (!!!)

So, what is the best strategy
for soluble queries ?

Square-Root Allocation

Consider a pair of items i, j : $q_i > q_j$; and parameterize the allocations by a variable x ($0 \leq x < 1$)

$$p_i = x (p_i + p_j), p_j = (1-x) (p_i + p_j)$$

Proportional \Rightarrow $p_i = q_i$, so $x = q_i / (q_i + q_j)$

Uniform \Rightarrow $p_i = p_j = \frac{1}{2}$

ESS $\propto q_i/x + q_j/(1-x)$ [inversely proportional to population]

(convex function) ESS equal for proportional & uniform

ESS is minimum when $x = \sqrt{q_i} / (\sqrt{q_i} + \sqrt{q_j})$

So, $p_i \propto \sqrt{q_i}$

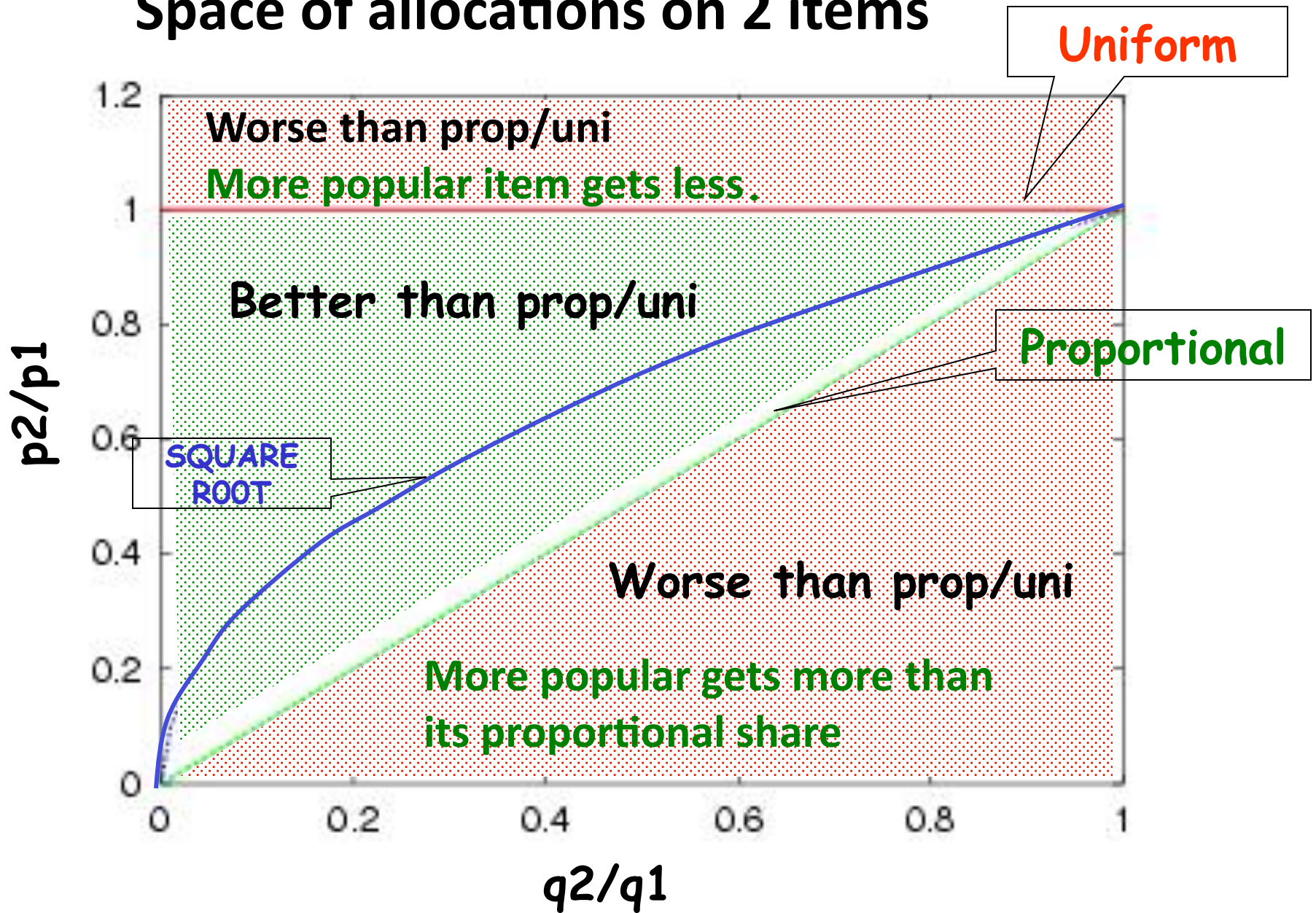
Square-Root Allocation

p_i is proportional to **square-root**(q_i)

$$p_i = \frac{\sqrt{q_i}}{\sum_{j=1}^m \sqrt{q_j}}$$

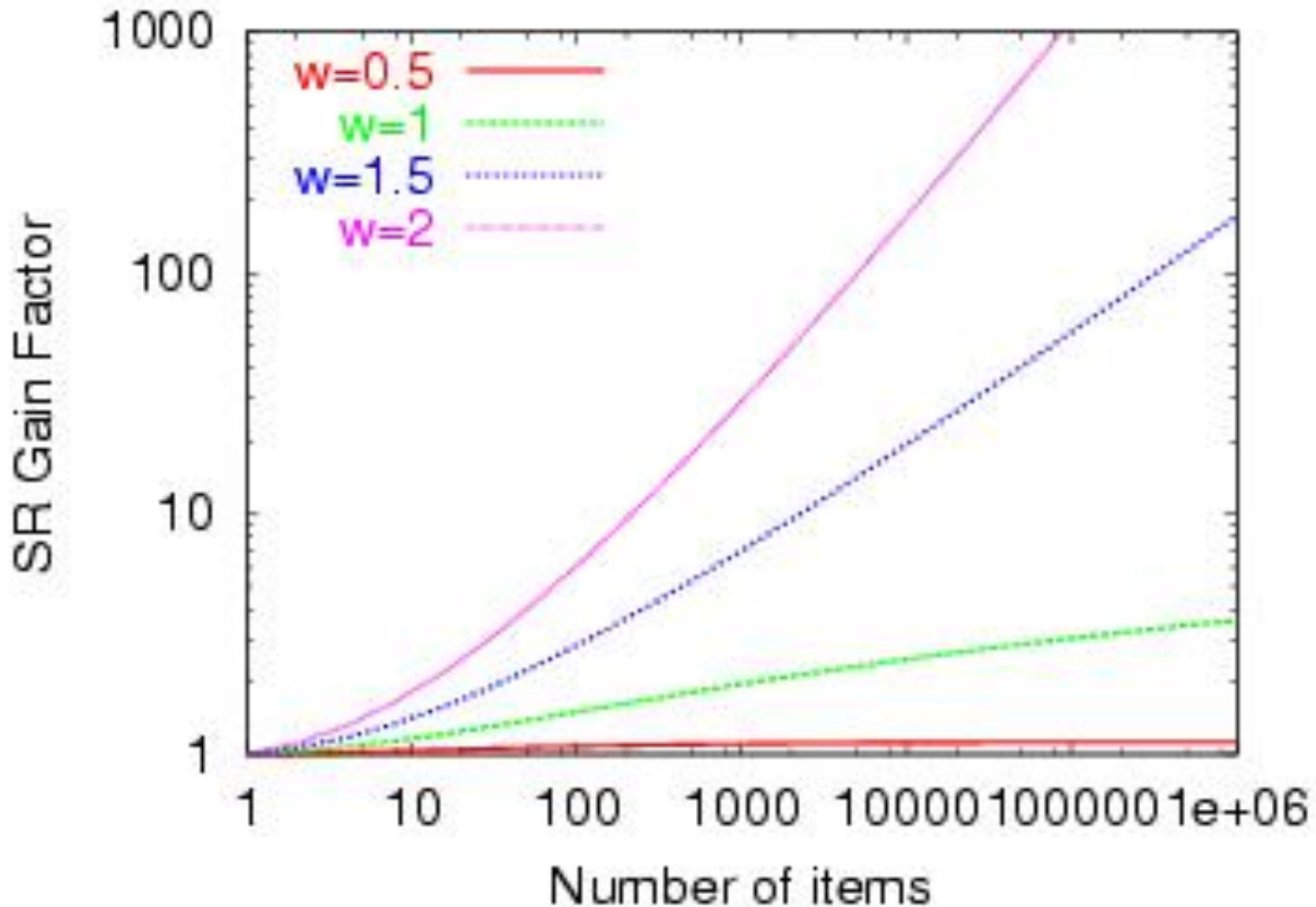
- Lies “In-between” Uniform and Proportional
- **Theorem**: Square-Root allocation minimizes the ESS (on soluble queries)

Space of allocations on 2 items



How much can we gain by using SR ?

Zipf-like query rates $q_i \propto i^{-w}$



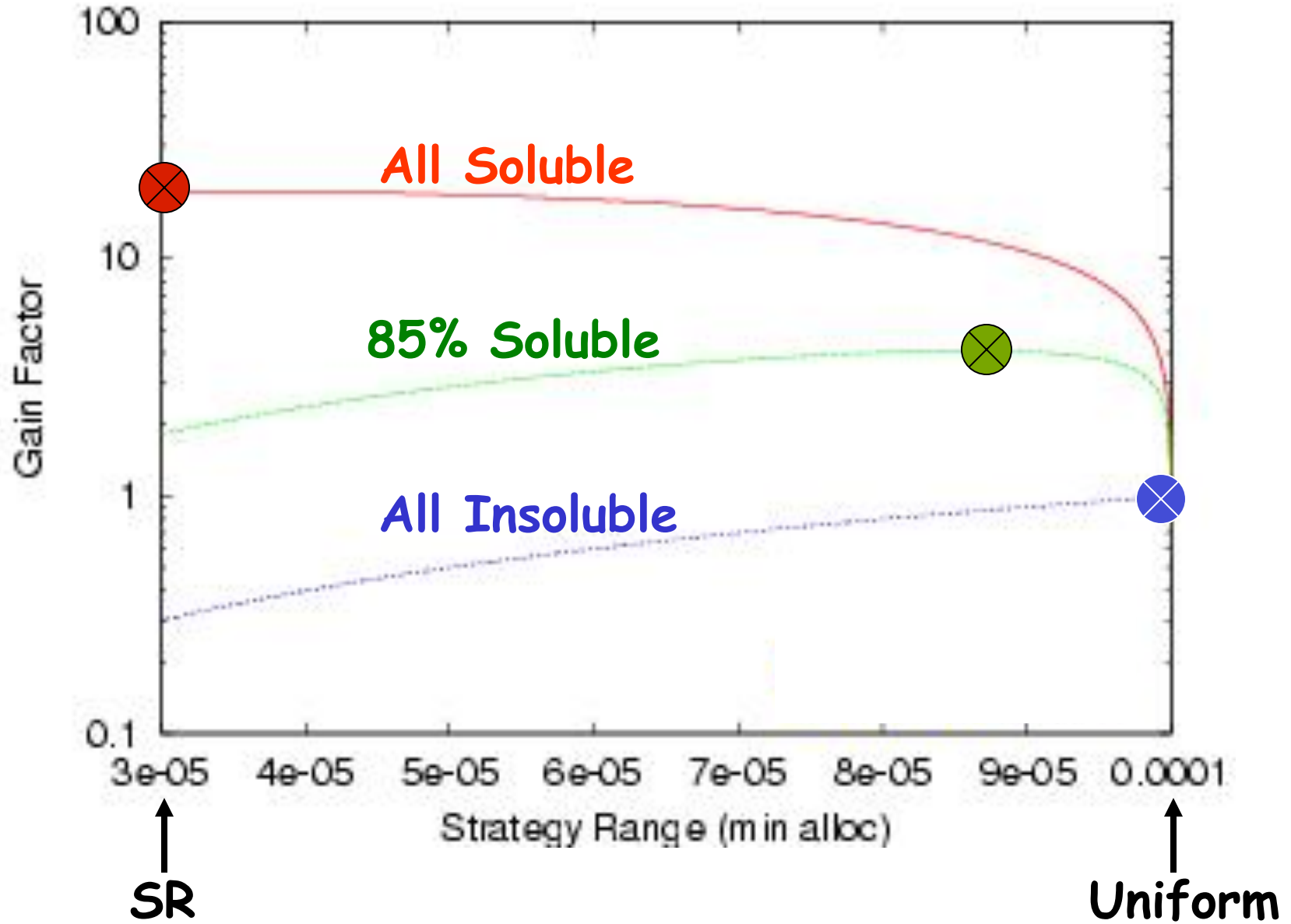
- **SR** is best for soluble queries
- **Uniform** minimizes cost of insoluble queries

What is the *optimal* strategy?

OPT is a hybrid of **Uniform** and **SR**

Tuned to balance cost of soluble and insoluble queries.

10⁴ items, Zipf-like w=1.5



Replication Algorithms

- Uniform and Proportional are “easy”
 - Uniform: When item is created, replicate its key in a fixed number of hosts.
 - Proportional: for each query, replicate the key in a fixed number of hosts

Desired properties of algorithm:

- Fully distributed where peers communicate through random probes; minimal bookkeeping; and no more communication than what is needed for search.
- Converge to/obtain SR allocation when query rates remain steady.

Model for Copy Creation/Deletion

- **Creation**: after a successful search of s , C_s new copies are created at random hosts.
- **Deletion**: is independent of the identity of the item; copy survival chances are non-decreasing with age. (i.e., FIFO at each node, per a TTL)

Property of the process:

$\langle C_i \rangle$ average value of C used to replicate i^{th} item.

Claim: If $\langle C_i \rangle / \langle C_j \rangle$ remains fixed over time, then

$$p_i / p_j \rightarrow q_i \langle C_i \rangle / q_j \langle C_j \rangle$$

Creation/Deletion Process

Corollary:

$$\text{If } \langle C_i \rangle \propto \frac{1}{\sqrt{q_i}} \text{ then } p_i/p_j \rightarrow \sqrt{q_i/q_j}$$

Algorithm for square-root allocation needs to have $\langle C_i \rangle$ equal to or converge to a value inversely proportional to $\sqrt{q_i}$


SR Replication Algorithms

- **Path replication**: number of new copies $C(s)$ is proportional to the size of the search (**Freenet**)
 - Converges to SR allocation (+reasonable conditions)
 - Convergence unstable with delayed creations
- **Sibling memory**: each copy remembers the number of sibling copies,
 - Quickly “on target”
 - For “good estimates” need to find several copies.

Algorithm 1: Path Replication

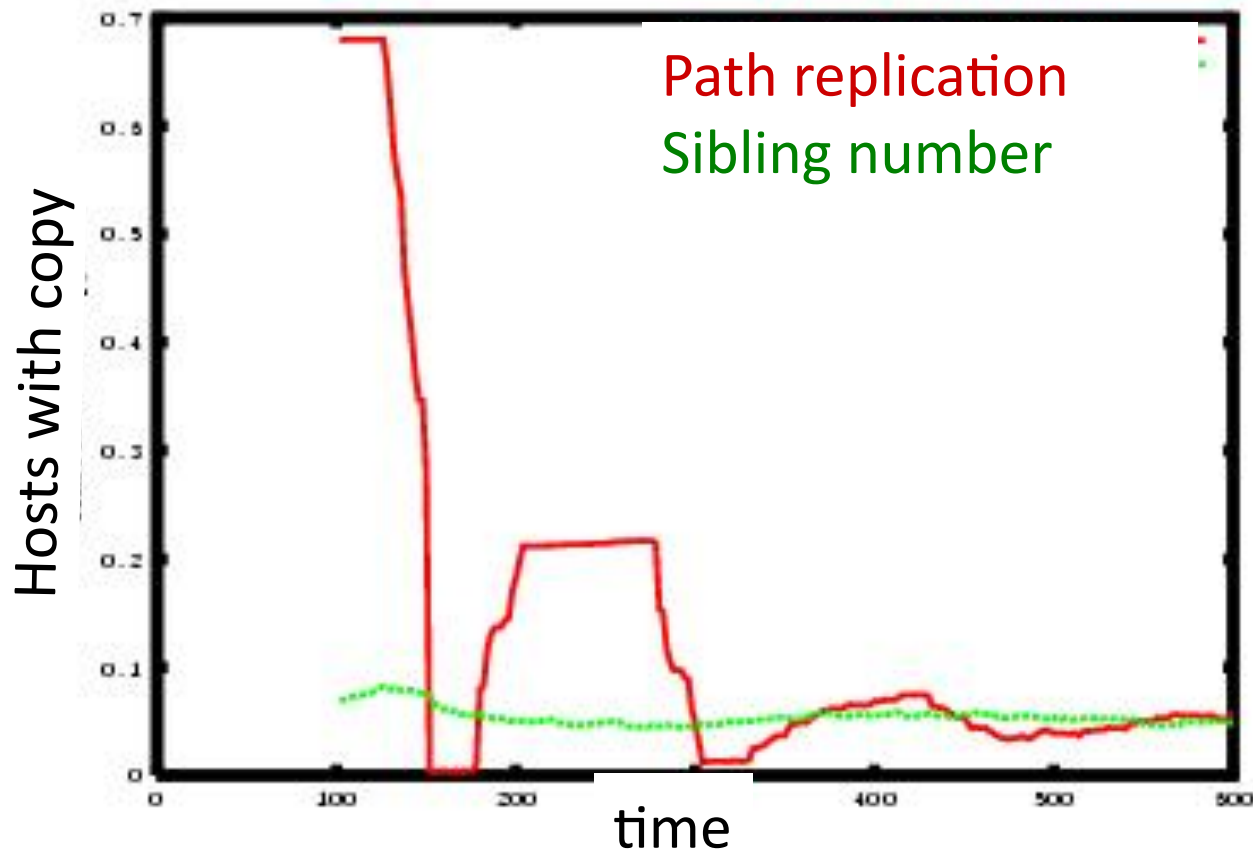
- Number of new copies produced per query, $\langle C_i \rangle$, is proportional to search size $1/p_i$
- Creation rate is proportional to $q_i \langle C_i \rangle$
- Steady state: creation rate proportional to allocation p_i , thus

$$q_i \langle C_i \rangle \propto q_i / p_i \propto p_i$$


$$p_i \propto \sqrt{q_i}$$

Simulation

Delay = 0.25 * copy lifetime; 10000 hosts



In this simulation there is delay of 25 time units in copy creation; the copy lifetime is 100 time units; and the inter-request time is 2.

Summary

- Random Search/replication Model: probes to “random” hosts
- Proportional allocation – current practice
- Uniform allocation – best for **insoluble** queries
- Soluble queries:
 - Proportional and Uniform allocations are two extremes with same average performance
 - **Square-Root allocation** minimizes Average Search Size
- OPT (all queries) lies between SR and Uniform
- SR/OPT allocation can be realized by simple algorithms.