1 Experiments

Data model:
Since we are dealing with mobile environments, we consider mobile navigation application as a representative of the domain. As our data model we consider a moving object (a person traveling in a car on land or taking the aerial route in distant future).This application has peculiar characteristics, such as linear motion (traveling on land along a national highway), circular motion, or sometimes even a random motion. Also, the speed of the motion can vary drastically; a person traveling slowly in a busy city with haevy traffic as compared to someone on a freeway with virtually no traffic. We model such a system by a relational spatial database. In this system, we consider a single relation that has some unique characteristics. Every tuple has two attributes, namely the X and the Y coordinate that represent a point in a 2-Dimensional space. For the sake of simplicity these attributes take only integer values. Tuples may contain additional attributes like school, restaurant, gas station, food joints, road segments, rivers, maps etc. that are associated with the X-Y coordinates.

Query Workload:
We can generate the query workload using a customized, pseudo-random generator. The desired query characteristics are specified and using the randomness different query streams are generated that all satisfy the desired characteristics. We can also set the selectivity of each query stream such that the resultset (number of tuples in the result) has a wide range. Queries are generated pseudo-randomly. i.e with different profiles, queries are varied accordingly. Example, movement on a highway is characterized with long term queries related to food joints, gas stations and rest areas. Whereas while moving in a city, people are often concerned with beating the traffic, finding the nearest parking garage, nearest laundry or consumer store or the nearest book store that has relatively fewer customers at a particular time. Queries implicitly include X-Y coordinates as a part of the select condition. For example a query could be " find me all restaurants in the vicinity" . The query is equivalent to saying " find me all restaurants in the vicinity where x=current-X and y=current-Y".

Goal of the Experiment:
The goal of this experiment is to find out whether our replacement heuristic is better than the traditional ones like LRU, MRU, FAR etc. To do so we test the cache effectiveness of the system. The cache effectiveness of a system is measured in terms of the time saved in answering a query locally as compared to answering it from remote server. This includes the time saved in executing the query at the remote server and the time required to transmit the query results back to the client.

Def: Response Time r_i : Time interval between the query request sent by the client and the entire resultset sent back to the client.

Goal:
To minimize the cost of servicing the requests that cannot be completely answered from the local cache.
Cost is measured in terms of time.
Factors that affect the cost are:

- Size of the data that need to be transfered from remote location(s). (Size of remainder query)
- Network parameters: Latency, bandwidth and transient conditions like server load, network congestion.
- Remainder query execution time

For an online algorithm we also need to take into account the Access probability of each query.
Access probability can be based on:

- frequency of access f
- freshness-count - a measure the determines how recently has the query been accessed.
- other domain specific paremeters (e.g. Semantic distance S_d - a parameters that indicates the distance between the moving object's current location and the location of the query issued by it)

Assumptions:
1. Response time is dependent on latency, size and bandwidth. Assuming uniform latency and bandwidth across the network setting, the response time will be dependent on size of the query.
2. We assume that hoverer computationally expensive, the query execution time is much smaller than the data transmission time. Hence we typically ignore the execution time while considering its impact on the response time.
With the above two assumptions we can deduce that the query response time is directly proportional to the size of the resultset.

Hence we come up with a cost function given by:
Cost function C = freshness-count + $(f * size)/S_d$

We use C as our primary metric, an indicator of how much time is saved by answering a query from cache. As a secondary metric, we could use the cache hit ratio, although this may not be the most accurate measure for cache effectiveness.

References