Traditional Pre-fetching and caching of limited use for mobile applications

Yann Michel[1], Annika Hinze[2]
[1] Freie Universitaet Berlin, ymichel@inf.fu-berlin.de
[2] University of Waikato, New Zealand, a.hinze@cs.waikato.ac.nz

Abstract: This paper reports our experiences introducing location and context aware caching and pre-fetching for a mobile tourist information system. An extensive evaluation and comparison of selected caching and pre-fetching strategies has been carried out in this mobile environment. We discuss why the consideration of location-awareness is less influential and thus less successful than expected and hoped for.

1 Introduction

The Tourist Information Provider (TIP) is a context-aware mobile tourist information system providing sight information to its users. The available information is filtered according to a user's location and interest profile. Our goal for the research reported in this paper was to lower the response times for mobile clients and to restrict wireless network usage to times of good network quality. We wished to alleviate or even eliminate the effects of disconnectedness so that the user can continue working with the TIP client even though network connection is (temporarily) unavailable. We wanted to retain TIP's characteristics of location-awareness and personalization.

As Kirchner et al. [6] pointed out, pre-fetching in combination with caching may lead to lower response time, based on more locally available data in the cache and less "burst" load on the network due to most of the information being downloaded at times with sufficient bandwidth. On the other hand one needs to consider additional cost at the mobile device: additional CPU load to predict the user's movements and possibly wasted bandwidth in case there was a falsely predicted movement.

This paper reports on our experiences introducing location and context aware caching and pre-fetching for the mobile TIP. We provide details of our extensive evaluation and comparison of selected caching and pre-fetching strategies in a mobile environment. The findings of our analysis are surprising; we will show that the consideration of location-awareness is less influential and thus less successful then expected and hoped for. Our conclusion is that additional information, similar to the one used in recommender systems, will need to be used to support more effective location-aware caching and pre-fetching.

The remainder of the paper is structured as follows: Section 2 discusses strategies for caching and pre-fetching applicable in a location aware mobile context; Section 3 introduces implementation details. The main focus of the paper is on the evaluation reported in Section 4. The details about the TIP system are kept to a minimum to allow for extensive discussion of the evaluation results. TIP has been extensively described, e.g., in [4].

2 Caching and Pre-fetching

Data in a *cache* is stored ready to be used and is replaced when the available space is used.

Replacement strategies suitable for location-based mobile systems Temporal replacement strategies consider timing information of the stored items, such as in *first-in first-out (FIFO)* and *least-recently-used (LRU)*. Semantic replacement in mobile applications could refer to location, such as in the *Manhattan Distance* and the use of *Voronoi Diagrams*. For the Manhattan Distance [3] all cached elements are assigned to a region; this replacement strategy removes all elements within a region and not just a single element (as in LRU). The region with the greatest Manhattan distance between its centre of gravity to the current point is replaced. Voronoi Diagrams [9] aim at accelerating nearest neighbor queries. Direction and current speed of the users is taken into account. Three replacement strategies are based on cached area, on distance, and on common circle pane. Spatial replacement strategies are optimized for location-aware item handling. Two typical strategies are *Furthest Away Replacement (FAR)* and *Probability Area/Probability Area Inverse Distance*. Furthest away replacement [8] uses direction prediction for the next estimated location. Elements to replace are those that are furthest away from the predicted position and that are not in the direction of movement. PA/PAID methods [10] additionally consider *data distance* and *valid scope area*. Data in the larger valid scope area have a higher request probability. The items with the lowest probabilities are replaced.

Pre-fetching Pre-fetching in Linux uses the LRU-One Block Lookahead (LRU-OBL) [5]. It pre-fetches the logical block next to the currently referenced block. The cache is partitioned into two rooms, the *Weighing Room* and the *Waiting Room*, where weights of the stored items are determined and the not yet referenced items are kept, respectively. A critical parameter is the optimal ratio for the sizing of the two rooms. Predictive pre-fetching of data in a mobile environment [2] uses a velocity model based on current location, speed, and direction of the user. For pre-fetching data for a user at a certain location, speed and the direction of the user have to be determined (resulting in a delay). Average speed of the mobile users may be used [6] to smooth the changes of the pre-fetching zone. The higher the user's speed, the larger the pre-fetching zone. In a mobile environment, the next movements of the mobile user must be predicted. Two typical approaches for estimating a user's future position are linear interpolation and Lagrange interpolation.

Combined Location-aware Caching and Pre-fetching: Cao et al. [1] have investigated how to efficiently use caching and pre-fetching without letting both interfere with each other. They propose four rules for optimizing the combination of caching and pre-fetching strategies and two combined usage strategies: A *conservative* strategy simply tries to minimize the elapsed time while performing a minimum number of pre-fetches; and an *aggressive* strategy that always pre-fetches the next block that is not in the cache at the earliest possible moment by replacing the block whose request is furthest away in the future. In our current implementation, we consider these rules by applying the conservative strategy.

3 Implementation

The TIP server is the central information provider for the available data; it consists of several services and a central broker that routes the information. The TIP client resides on the mobile devices; it is the end user software component for travellers. The client consists of a broker and several services; it runs a cache and a pre-fetching service. A *Message Handler* maintains the cache and the stored cache items. If all replies to the incoming request are already cached, they are simply enqueued into the reception queue. If not all replies are locally available, the request is forwarded to the remote broker. Requests are handled as events. An incoming event is entered into the cache and then enqueued into the reception queue. This cache implementation provides an abstract framework for all cache methods used in TIP. As internal container for the cached objects we use a HSQL database[1] that runs in memory or by using a local storage such as a compact-flash or hard-disk. We keep two tables EVENTS and REPLIES. To support location-aware filtering of the stored events, several geo-spatial functions were added to the database. We analysed the following cache replacement policies for the TIP client: LRUCache (LRU replacement strategy) MHDCache (manhattan distance), FARCache (distance and direction).

The system pre-fetches data for locations the user is expected to move to, using the *PrefetchService*. The system distinguishes between prefetched and cached events by different markings: *cached* events were previously requested by the user, e.g., by a GPS-triggered Location event; *pre-fetched* events were requested by the submission of a predicted location. This results in a two-room scheme similar to the one introduced earlier. For the movement prediction of the TIP user, we use linear and Lagrange based prediction. The *PrefetchService* advertises *Location* events and subscribes to all events the *DisplayService* subscribes to. When a user arrives at a certain location and the cache was appropriately sized, the desired information should be locally available already and does not need to be specifically requested from the server.

4 Performance Analysis Setup

We start this central section by describing the evaluation parameter and measurements selection, and the setup of the tests.

4.1 Evaluation Parameters

Cache size: The cache size defines the number of items that can be stored in the cache. Here we do not specify the overall storage size used but the maximum number of elements that can be stored. This simplification of the cache sizing was chosen because the implemented sample services submit events of similar size. We test caches with 250, 500, 1000 and 2000 elements.

Cache replacement strategy: Three cache replacement strategies can be chosen for TIP: least recently used (LRU), manhattan distance (MHD) and furthest away replacement (FAR) as described in Section 2

User location: This parameter specifies the location of the TIP user. We test three scenarios: triangle, circle and random (see Section 4.3).

[1] http://www.hsqldb.org/

Pre-fetching strategy: Different future user locations are predicted using "linear pre-fetching" and "Lagrange pre-fetching" with 3, 4, or 5 historic location points, and the disabling of pre-fetching. This predicted location is submitted to the pre-fetching service by event message.

Cache retrieve distance ϵ: The implemented cache allows the retrieval of similar events that are within a distance of ϵ degrees. Here this value is set to zero to test only the underlying cache replacement strategies and not the retrieval of similar events.

4.2 Evaluation Measures

The evaluation focussed on four measures. The first three measures were probed inside the message handler. If an event is a request, the handler checks if the requested information is still cached (if yes, this is a cache hit, otherwise a cache miss).

Cache Hit-ratio The cache hit-ratio is a quality measure for the cache replacement strategy. A high hit-ratio indicates a good re-usage of cached items and therefore a good quality of the underlying cache replacement strategy. A low hit-ratio shows that less stored items could be re-used. A high cache hit-ratio indicates less network requests and therefore less energy consumption; a low cache hit-ratio indicates a larger amount of data that needs to be fetched via the network.

$$\textit{cache hit-ratio} = \textit{cache hits}/(cachehits + \textit{cache misses})$$

Pre-fetch Hit-ratio If the event that is returned from the cache has the flag *pre-fetched* set to true, this is counted as a pre-fetched event. The flag is now changed to false and the event is returned. If the event was not pre-fetched, it is only counted as a returned event. A high pre-fetch hit-ratio implies that the pre-fetching service already downloaded the desired data; a low pre-fetch hit-ratio does not necessarily mean that the algorithm performs poorly but could also be the result of a high pre-fetch delete-ratio.

$$\textit{pre-fetch hit-ratio} = \textit{number of pre-fetched events}/\textit{number of all events}$$

Pre-fetch Delete-ratio When the cache is full and a new item should be inserted, an already stored item has to be removed. If the determined victim still has the pre-fetched flag set, it was never accessed by the user. We count the deletion of all items, plus the items that were pre-fetched but newer accessed by the user. A low pre-fetch delete-ratio indicates that much of the pre-fetched information was really accessed and therefore used. A high pre-fetch delete-ratio indicates that unnecessary downloads were generated and results in a waste of energy. The pre-fetch hit-ratio together with the pre-fetch delete-ratio show how effective the evaluated pre-fetching strategy works with the used cache replacement.

$$\textit{pre-fetch delete-ratio} = \textit{number of deleted pre-fetched events}/\textit{number of all deleted events}$$

Prediction Distance The distance of the predicted location to the actual location measures the quality of the movement prediction. A low prediction distance is a good quality measure for the underlying algorithm. If the prediction distance is high it is likely that the desired information has not been cached on arrival at a future location. The quality measure for the predicted locations is probed inside of the pre-fetch service, which stores the last predicted location.

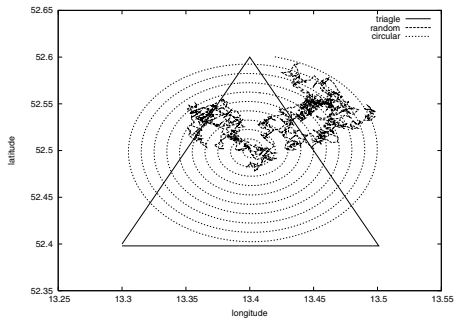


Figure 1: Example tracks for the test scenarios.

4.3 Evaluation Setup

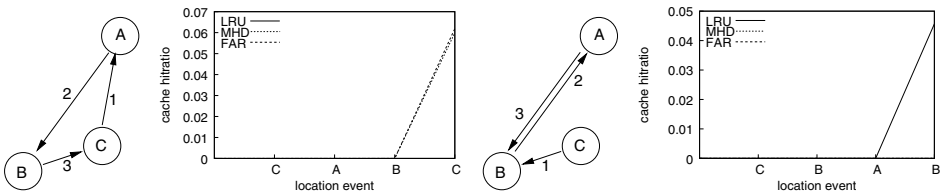
We give an overview of the sample data used and give details about the test scenarios that were used to evaluate the system.

Sample Data The evaluation data are artificially generated but influenced by observed real data patterns. First we generated sample data for sights that can be visited and contains at least a name and corresponding coordinates. The sample-sights are uniformly distributed in the geo-spatial surrounding of Berlin: 13.3 to 13.5 degrees east and 52.4 to 52.6 degrees north. Distance between sights in longitudinal and latitudinal direction is 0.001 degrees (approximately 111m) each. 27,647 sample sight entries were generated.

Discussion of suitability of data and tracks We use artificial test data that are uniformly distributed(as described above) and a number of test scenarios: a forces scenario, a circular track, a triangle track, and a random track (described below). These tracks are somewhat restricted and, taken in isolation, unrealistic mobility models to simulate a user' movement. However, we believe even using these tracks will allow us to make interesting observations. After all, a user's movements will consist of a sequence of movements akin to the patterns evaluated here. We focus on situations where a user follows a seemingly random sequence of these movements. That is, no information about the terrain or other users' walking patterns is available. We will discuss later which improvements could be used if more information is available. We believe that our systematic approach of evaluating walking patterns that may in combination describe a user's track will allow for a general analysis of the evaluated algorithms.

Test Scenarios For the evaluation of the described parameters, three test scenarios were defined. Each of them describes a track (see Figure 1) that a virtual user traverses:

S1. Triangular Track: This track describes a linear movement of a user with two turning points at the corners of the triangle. Lines one to five generate the rising edge; lines six to ten generate the falling edge; lines eleven to fourteen generate the edge that returns to the start. Offset parameters specify the step size in longitude and latitude direction.



(a) Track 1 (b) Track 1 hit-ratio (c) Track 2 (d) Track 2 hit-ratio

Figure 2: Example Tracks 1 and 2 for demonstrating the different behavior of location-aware and -unaware cache-replacement strategies shown in the hit-ratio graphs.

S2. Circular Track: This track describes a spiral movement of the user. The parameters δt and *magnitude* specify the sampling resolution and the distance between the drawn spiral-line respectively. Here, the offset parameters for longitude and latitude move the generated spiral to the center of the pane that was filled with the test data.

S3. Random Track: This track describes a randomly chosen track. Generated locations on the defined pane are kept, otherwise discarded. A multiplicator is randomly generated between -0.5 and 0.5. The offset parameters specify the factors in longitude and latitude direction that are used for the computation of the next location.

For the evaluations tracks were generated once and used for every test setup.

5 Performance Evaluation Analysis

For each test we first describe our hypothesis and then compare these to the results and discuss the significance of the results.

5.1 Caching

We first demonstrate the advantages and disadvantages of location-aware caching in a forced scenario. We then test the cache replacement strategies (LRU, MHD and FAR) for each test scenario and different sizes.

5.1.1 Forced Scenario

This test compares the behavior of the location-aware and location-unaware cache replacement strategies. We used two tracks created to be favorable or unfavorable, respectively, for each of the two categories. This scenario will highlight how different caches behave differently under different circumstances depending on the replacement strategy and the track. Each track consists of four location visits (see Figures 2(a) and 2(c)). To force the replacement of stored items, the cache size is restricted to three locations.
The results for Track 1 are shown in Figure 2(b). The location-aware algorithms both

have replaced items for region "A" because the distance to the stored items was furthest away from the current location. The items for the region "C" were retained resulting in a cache-hit when returning to that region. Figure 2(d) shows the results for Track 2. The location-aware algorithms replaced region"B" because it was furthest away from the current location. Consequently, when returning to "C" the items were no longer locally available and had to be re-fetched. The LRU-algorithm replaced the items for the region "C" resulting in a cache-hit when returning to "B". These results give rise to the expectation that the time-aware algorithms perform better for "returning" users whereas the location-aware algorithms performs better for users moving in "circles".

5.1.2 Test Scenario 1 (Triangle Track)

We expected the LRU- and the MHD-based caches to perform similarly because, except for the two turning points, the track describes a linear movement. The time-based replacement of LRU algorithm replaces the same items as MHD based on distance. The FAR-based algorithm is expected to perform similar until the first turning point has passed. Then we expect a temporary decrease in performance since it does not only consider the distance of the items but also the movement direction.

Only one example result is shown in Figure 3(a); as the results were almost identical for all four cache sizes tested (250, 500, 1000, 2000). Only cache size 1000 had a slight divergence in the hit-ratio for the FAR-based cache: The cache hit-ratio decreases after the first turning point of the triangle track but then shows a slightly increasing trend afterwards. The reason is that the victim chosen at the first turning point is disadvantageous to the cache hit-ratio.

Overall, the algorithms performed as expected. However, it was surprising to find that the cache size does not have any significant effect on the hit-ratio of the evaluated algorithms.

5.1.3 Test Scenario 2 (Circular Track)

For all four cache sizes and all algorithms, we expected the cache hit-ratio to decrease over the test track because the user walks from the center of the spiral outwards.

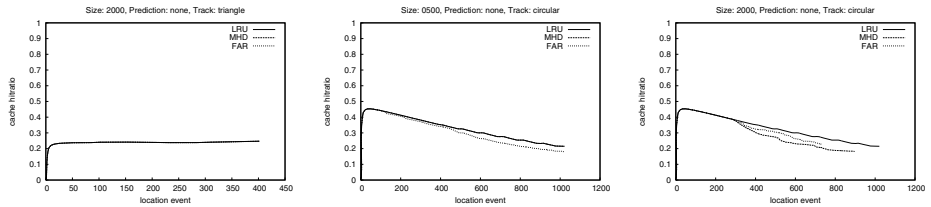
Two results are shown in Figure 3(b) and 3(c); different cache sizes resulted in almost identical results for the LRU-based cache. The MHD-based cache performs identically to the LRU-based algorithm for a cache size of 250/500 elements, with only slight changes for 1000 and 2000 elements. The FAR algorithm has the lowest cache hit-ratio for all sizes. The results for 250 elements are similar to the other caches; the hit-ratio decreases with increasing cache size.

The algorithms performed as expected. The low hit-ratio of the FAR-cache is caused by the direction determination. The larger the radius of the circular walk, the more items are cached and need replacement while moving along the circle. After a threshold, recently visited items have to be replaced first, leading to less items being available when returning to the same angle of the circle.

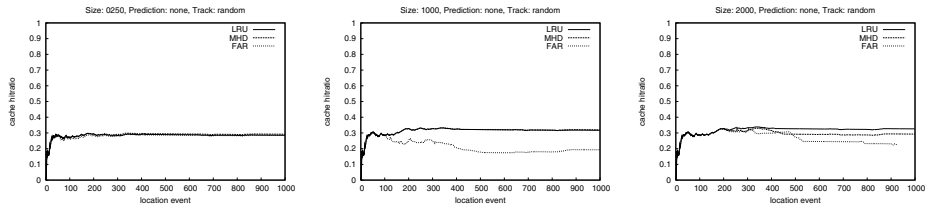
5.1.4 Test Scenario 3 (Random Track)

We expect the location-aware caches (MHD and FAR) to perform better than the location-unaware cache (LRU). We also expected larger cache sizes to improve the cache hit-ratios.

Selected results are shown in Figures 3(d) to 3(f). The MHD-based cache performs slightly better than the LRU- and FAR-based caches for 250 elements. When increasing the



(a) Triangle: 2000 (b) Circular: 250 (c) Circular: 2000



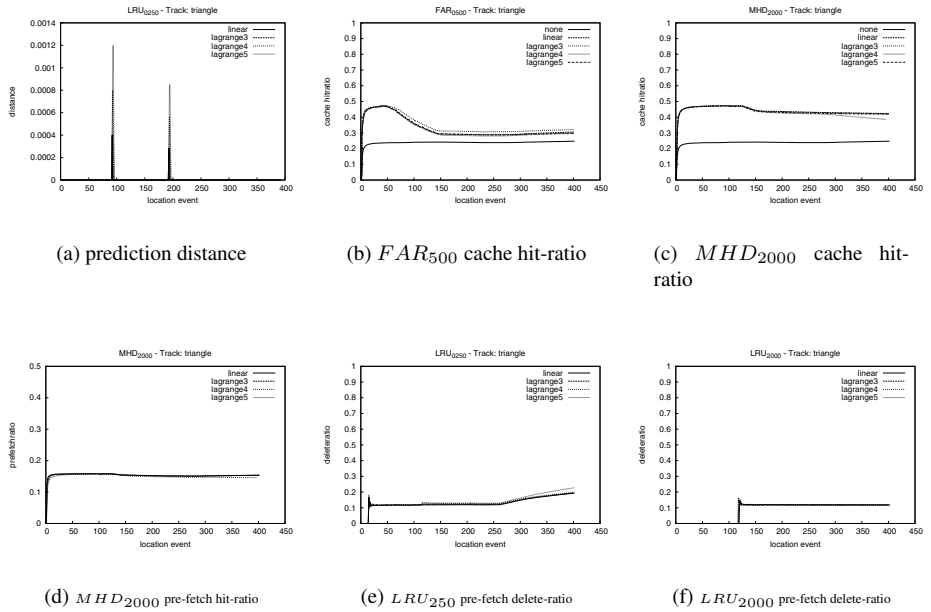
(d) Random; 250 (e) Random: 250 (f) Random: 250

Figure 3: Cache hit ratio for Scenario S1, S2, S3 (track type: cache size)

cache size, the LRU cache hit-ratio increases (LRU-based cache has the best hit-ratio for 1000/20000). Figure 3(e) shows decreasing ratio for the FAR-based cache. For 2000 elements (Figure 3(f)), both location-aware caches perform worse than the LRU-based cache. Not all our expectations were met in this scenario. The bigger cache size only improved the LRU- and the MHD-based caches whereas it worsened the FAR-cache's results. The reason lies in the different replacement in FAR compared to MHD: FAR takes distance and movement direction into account. In addition, increasing the cache size seemed to improve only the LRU-based cache. Based on the similar graphs shapes, one can see that the basic character of the distance-based replacement (MHD) is similar to the time-based replacement (LRU). The location-aware algorithms performed best for the small sizes – for larger cache sizes, the LRU-based algorithm has more space to store its cached items.

5.1.5 Summary: Caching

We have shown that the system benefits from caching: 15 to 45 percent of the requests could be answered locally by re-using cached information. In these cases the mobile client did not need to re-download the information via the network, resulting in energy savings for network access. The effect of location-aware caching was not as significant as expected: For small cache sizes, the results of the location-aware (MHD and FAR) caches were largely comparable to those of the location-unaware cache (LRU). For larger cache sizes, the LRU-based cache replacement results in better hit-ratios than the other algorithms.



(a) prediction distance

(b) FAR_{500} cache hit-ratio

(c) MHD_{2000} cache hit-ratio

(d) MHD_{2000} pre-fetch hit-ratio

(e) LRU_{250} pre-fetch delete-ratio

(f) LRU_{2000} pre-fetch delete-ratio

Figure 4: Selected pre-fetching results for Scenario S1 (triangle track).

5.2 Pre-fetching

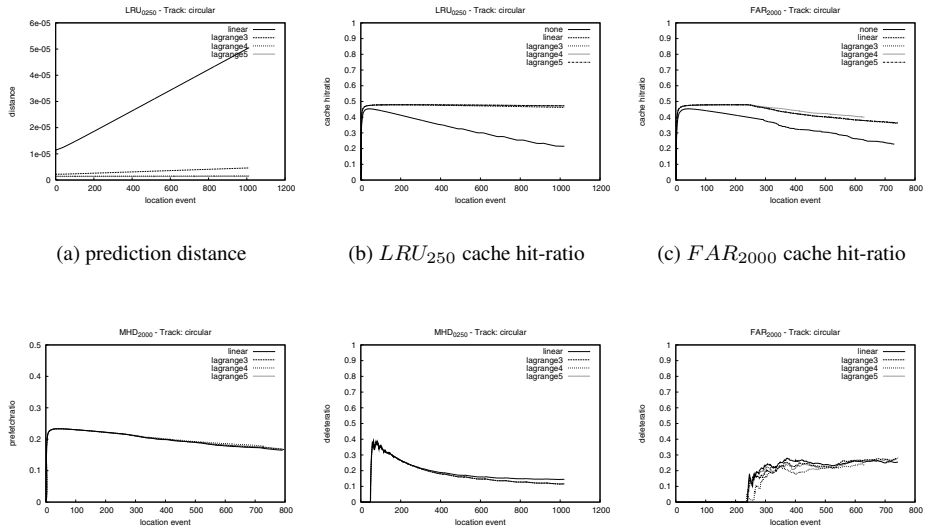
In this subsection, we discuss the results of the pre-fetching algorithms, using the the same tracks and cache sizes as before.

5.2.1 Test Scenario 1 (Triangle Track)

The prediction distance for all implemented methods should be zero for linear movements. We expected all algorithms to perform similarly independent of the replacement strategy. We predicted slightly better cache hit-ratio for larger cache sizes. We show selected results that are indicative of the observed pre-fetching behavior.

1. Prediction distance: The prediction distance is the same for all replacement strategies and not affected by the cache size. Figure 4(a) shows the results for the LRU cache with 250 elements. Except for the mis-prediction at the two turning points, the prediction distance is always zero.

2. Cache hit-ratio: Introducing pre-fetching leads to a higher hit-ratio; all pre-fetching strategies lead to identical hit-ratio graphs for each selected cache replacement strategy and cache size (see Fig 4(b) and 4(c)). After a drop-off point, the hit-ratio decreases. At this point the cache is full and items have to be replaced. The decrease continues until a plateau is reached. Drop-off and plateau point are different for different caching strategies and cache sizes.



(a) prediction distance

(b) LRU_{250} cache hit-ratio

(c) FAR_{2000} cache hit-ratio

(d) MHD_{2000} pre-fetch hit-ratio

(e) MHD_{250} pre-fetch delete-ratio

(f) FAR_{2000} pre-fetch delete-ratio

Figure 5: Selected pre-fetching results for Scenario S2 (circular track).

3. Pre-fetch hit-ratio: The different pre-fetching algorithms resulted in similar pre-fetch hit-ratios for the same cache size. An example is shown in Figure 4(d). A difference could only be measured for the FAR-based experiments of the size 500 and 1000 where we observed a slight decrease in the pre-fetch hit-ratio to 12 percent after the caches were filled.

4. Pre-fetch delete-ratio: Similar to the cache hit-ratio, pre-fetch delete-ratio is influenced by the cache replacement strategy and the cache size. Examples are shown in Figures 4(e) and 4(f). The delete ratio starts low and sharply increases at some point to stay relatively stable but higher afterwards. The results vary slightly for different location-prediction algorithms.

The evaluated pre-fetching strategies did not show any significant differences. The increase of the cache size did not increase the cache hit-ratio as expected but delayed the replacement of stored items. Therefore the decrease of the cache hit-ratio of the MHD and FAR caches was delayed. In addition, the increased size of the cache decreased the pre-fetch delete-ratio for the underlying caches. The pre-fetch delete ratio indicates that a stable percentage of items were pre-fetched but never used.

5.2.2 Test Scenario 2 (Circular Track)

We expected the prediction distance for the Lagrange-based algorithms to be better than the linear prediction because of better adaptation of the polynomial to the circular track.

The prediction distance is expected to improve especially if the prediction algorithm uses more historic locations (higher Lagrange index). An increase of the cache size should not increase the cache hit-ratio because the track never returns to the same location.

1. Prediction distance: The prediction distance is not affected by replacement strategies nor the cache size. Figure 5(a) shows the results for the LRU 250 cache. The Lagrange-based prediction methods perform better than the linear prediction. All prediction methods show only a very small difference from the predicted location to the actual one (at most $5^{-5} degrees = 5m$).

2. Cache hit-ratio: Different pre-fetching methods do not result in significantly different cache hit-ratios. A result for smaller caches is shown in Figure 5(b)): the hit-ratio remains stable. For larger caches, the cache hit-ratio decreases. The actual shape of the graph is influenced by the replacement strategy; an example is shown in Figure 5(c)).

3. Pre-fetch hit-ratio: The different pre-fetching algorithms resulted in similar pre-fetch hit-ratios for the same cache size; an example is shown in Figure 4(d).

4. Pre-fetch delete-ratio: The pre-fetch delete-ratios for smaller caches are similar for all replacement strategies: the delete-ratio decreases after the cache is filled. Slight variations can be found for the FAR cache and for different prediction algorithms (see Figure 5(e)).

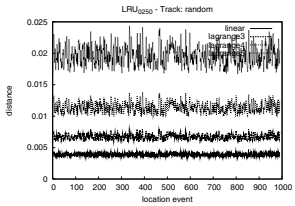
   With larger cache sizes, the delete-ratio graphs change: it is initially zero and sharply rises at a point to about 22 percent and from then on it continues to increase slowly with variations. The exact shape depends on the pre-fetching algorithm. The pre-fetching algorithm shows a wider variation for the FAR cache, especially at the start of the deletions, where we could measure a difference of up to 10 percent.

The prediction distances were very small. The Lagrange-based algorithms performed better than the linear algorithm especially when using more locations. These comparatively small differences between the prediction algorithm are reflected in the minimal difference of the cache hit-ratios. Surprisingly, an increased cache size decreased the hit-ratio for the FAR and MHD caches and the used pre-fetching algorithms. The decreasing cache hit-ratio is caused by an increased pre-fetch delete-ratio, i.e., the victims are disadvantageously chosen. This effect is amplified while increasing the cache size. The LRU cache, however, was not affected by the increased cache size.

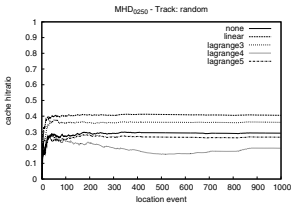
5.2.3 Test Scenario 3 (Random Track)

We expected this test to perform similarly to the one without pre-fetching: for the location-aware caches (MHD and FAR) to perform better than the time-aware cache (LRU). We also expected a better prediction accuracy and therefore a higher cache hit-ratio whenever a test setup uses a Lagrange-based prediction rather than the linear prediction. Generally, we expected a larger cache size to improve the cache hit-ratio measures.

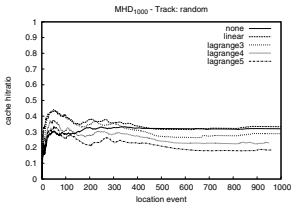
1. Prediction distance: The measurements for the prediction distance are the same for all experiments and did not change for any used cache replacement strategy or cache size (see example in Figure 6(a)). The prediction distance of the linear prediction method is approximately 400 m. The prediction distance of the Lagrange methods using three, four and five points are 600 m, 1.1 km and 2 km, respectively.



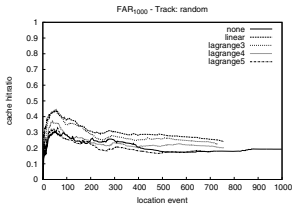
(a) prediction distance



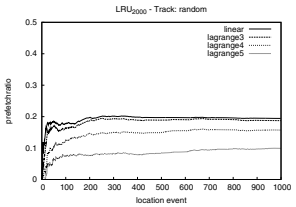
(b) MHD_{250} cache hit-ratio



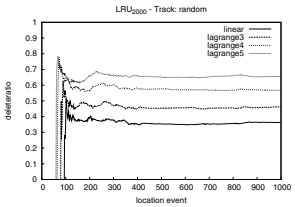
(c) MHD_{1000} cache hit-ratio



(d) FAR_{1000} cache hit-ratio



(e) LRU_{2000} pre-fetch hit-ratio



(f) LRU_{2000} pre-fetch delete-ratio

Figure 6: Selected pre-fetching results for Scenario S3 (random track).

2. Cache hit-ratio: For the smaller caches, all algorithms show the best hit-ratios for the linear and Lagrange3 pre-fetching (see Figure 6(b)). An increased cache size of 500 elements improved the cache hit-ratios of the LRU and MHD caches. For the FAR cache, all pre-fetching algorithms resulted in a lower cache hit-ratio than without pre-fetching. Results for Lagrange with 4 and 5 points improve with increased cache size. However, the MHD cache results are worse than the non pre-fetching cache hit-ratio (see Figure 6(c)). The FAR cache shows the best results for the linear pre-fetching (see Figure 6(d)). The largest experiment size of 2000 storable elements did not improve the results for the LRU cache. The results of the linear and lagrange3 pre-fetching using the MHD or FAR cache were only slightly better than the results of the non pre-fetching experiment.

3. Pre-fetch hit-ratio: The pre-fetch hit-ratio can be improved for all caches by increasing the size. The best pre-fetch hit-ratios can be achieved for the linear pre-fetching (see Figure 6(e)) followed by the Lagrange-based algorithms.

4. Pre-fetch delete-ratio: The pre-fetch delete-ratio using LRU or MHD can be reduced by increasing the cache size. For the FAR cache, this only applies for the linear and lagrange3 pre-fetching algorithms. Lagrange with 4 and 5 points using an FAR cache do not show clear results. An example for the general pre-fetch delete-ratio of the used pre-fetching algorithms is shown in Figure 6(f).

Prediction distance for the linear algorithm was the smallest for all experiments. Against our expectations, the Lagrange-based algorithms did not show the smallest distance for the prediction, which may have be caused by the noise in the Lagrange polynomials generated. Only the three point Langrange-interpolation came close to the prediction distance of the linear algorithm. Although the cache hit-ratio and pre-fetch hit-ratio for the LRU_{2000} have similar results for the linear or lagrange3 pre-fetching (see Figure 6(e)), the pre-fetch delete-ratio for the lagrange3 algorithm is significantly higher (see Figure 6(f)). The cache size did not improve the cache hit-ratio, however, it decreased the pre-fetch delete-ratio for the LRU and MHD caches.

5.2.4 Summary: Pre-fetching

We have shown that the overall cache hit-ratio can be improved by pre-fetching data. We highlighted that this is not always the case, e.g., for the FAR_{500} experiment using the random track, the resulting cache hit-ratio was lower with pre-fetching than without. The linear prediction performed best for the random track; the Lagrange-based algorithms had the lowest distance on the circular track; all prediction algorithms performed the same on the triangle track. The pre-fetch hit-ratio did not show any significant differences for the different pre-fetch algorithm on the triangle and circular tracks. For the random track, the pre-fetching algorithms generated different results: The hit-ratio of the linear pre-fetch did perform best for all cases, with langeange3 algorithm closest in quality. The pre-fetch delete-ratio for the triangle and circular scenarios did not show any significant difference for the used pre-fetching algorithms. Again, for the random track, the best results were achieved by using the linear prediction. The size of the underlying cache did not significantly influence the cache hit-ratio of the linear or circular tracks. For the random track the cache hit-ratio was slightly improved by an increased cache. In addition, the pre-fetch delete-ratio could be lowered by increasing the cache size.

6 Concluding Discussion

This paper reports about an extensive evaluation of an location-aware cache and pre-fetching service implemented for a mobile tourist information system. Selected results were discussed, more details are available in [7].

We showed that the cache performance not only depend on the replacement strategies used but also on the user movements. Using a pre-fetching service, we further improved the results of the cache hit-ratio. However, we note that pre-fetching information can decrease the performance when the wrong replacement items are chosen. We also determined that not all pre-fetched information was used and was therefore downloaded unnecessarily.

The overall results of the performance evaluation can be summarized as follows: Caching is generally beneficial for the mobile system resulting in less downloads and therefore less energy consumption. Pre-fetching of information can additionally improve the cache hit-ratio but false-positives may occur, i.e., unnecessary download of items that are never used. Surprisingly, the simple cache replacement algorithm (LRU) and the simple pre-fetching algorithm (linear) performed best or at least competitively in comparison to the other algorithms. As a next step we plan to pre-fetch information depending on the available network quality and test with tracks that reflect this feature. We also observed how the cache performance depends on the scenarios: For a future evaluation, one could adapt the replacement strategy during runtime.

What are the benefits of this evaluation? The analysis was done in a systematic way, not depending on particular walks in a given city but by evaluating the building blocks of a user's track: linear, circular, and random movements. As outlined before, we assumed here that no information about the area, or the (typical) movements of other users is available. We found that caching and pre-fetching strategies that are traditionally employed do not work well in these settings. Some of the issues were anticipated but the extend to which these traditional algorithms are unsuitable is much larger than hoped for.

We conclude that effective caching and pre-fetching in a mobile location-aware environment has to use additional information to improve the quality. Candidates are (1) maps or other information about a city layout, that is, information tracks available to the user at any given location, (2) information about typical user behaviour – usage statistics for the tracks known from the map, and (3) information about *this* user's behaviour - that is, usage statistics that reveal preferences defined, e.g., by work and home location. Techniques similar to recommender strategies may be used to create a specific location profile for a user that allows movement prediction in locations the user has not visited before.

Security and Privacy issues were not targeted so far. Also, currently the client cannot verify the validity of the received information, which is important for usage in P2P networks.

References

[1] P. Cao, E. W. Felten, A. R. Karlin, and K. Li. A Study of Integrated Prefetching and Caching Strategies. In *Proc. ACM SIGMETRICS Performance Evaluation Review*, May 1995.

[2] G. Cho. Using Predictive Prefetching to Improve Location Awareness of Mobile Information Service. In *Proceedings of the ICCS'02*, London, UK, 2002.

[3] S. Dar, M. J. Franklin, B. T. Jónsson, D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. In *Proc. of the 22th VLDB*, Sept. 1996.

[4] A. Hinze and G. Buchanan. The Challenge of Creating Cooperating Mobile Services: Experiences and Lessons Learned. In *Twenty-Ninth Australasian Computer Science Conference (ACSC 2006)*, Hobart, Australia, Jan. 2006.

[5] H. S. Jeon and S. H. Noh. Dynamic Buffer Cache Management Scheme Based on Simple and Aggressive Prefetching. In *Proceedings of the 4th Annual Showcase And Conference (LINUX-00)*, Berkeley, CA, Oct. 2000. The USENIX Association.

[6] H. Kirchner, R. Krummenacher, T. Risse, and D. Edwards-May. A Location-aware Prefetching Mechanism. In *Proc. of 4th International Network Conference (INC 2004), University of Plymouth, Plymouth*, 2004.

[7] Y. Michel. Location-aware caching in mobile environments. Master's thesis, Freie Universität Berlin, Department of Computer Science, June 2006.

[8] Q. Ren and M. H. Dunham. Using Semantic Caching to manage Location-dependent Data in Mobile Computing. In *Proceedings of MobiCom'00*, New York, USA, 2000.

[9] B. Zheng and D. L. Lee. Semantic Caching in Location-Dependent Query Processing. In *Proc. of the 7th International Symposium on Advances in Spatial and Temporal Databases*, Redondo Beach, CA, USA, July 2001.

[10] B. Zheng, J. Xu, and D. L. Lee. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. *IEEE Trans. Comput.*, 51(10):1141–1153, Oct. 2002.