Mobile Information Systems 1 (2005) 91–108
IOS Press

Data retrieval for location-dependent queries in a multi-cell wireless environment

James Jayaputera and David Taniar
School of Business Systems, Monash University, Clayton, Vic 3800 Australia
E-mail: {James.Jayaputera, David.Taniar}@infotech.monash.edu.au

Abstract. The advances of mobile computing technologies in recent years contributes to the growing of mobile information services. Consequently, we need to give special attentions to producing correct answers of those requests, considering the scope of queries and user locations. In this paper, we propose an approach of mobile query processing when the users location moves from one Base Station to another and the queries cross multi-cells. The efficiency of our proposed approach is presented by giving some examples and performance evaluations.

1. Introduction

Location-Dependent Information Service (LDIS) is one type of applications to generate query results based on location of users issuing the queries [1,2,14,15,17]. This implies that, whenever the users change their locations while they are sending queries, the query results have to be based on the receiving location of the users' queries. Location-Dependent Query (LDQ) is one type of queries based on the data found on that a particular location [6,17]. Hence, the expected results of LDQ must be accurate and based on the new location of users. For example, a mobile user might issue a query such as: "List all ATM machines within a radius of 1 km from my current location". The results given for this query depend on the latest location of query issuers.

Location, either for users or targets, can be defined in two-dimensional coordinates (2D) or three-dimensional coordinates (3D) [11]. The 2D coordinates is a coordinate formed by *x*-axes and *y*-axes. On the other hand, the 3D coordinates is a coordinate formed by *x*, *y* and *z* axes. To simplify our discussion, we define the term *location* as two-dimensional coordinates, as we can map the 3D into 2D coordinates.

When the users move from one location to another, the new location can be the closest or furthest area from the current one. In a mobile computing environment, a certain area is controlled by a Base Station (BS). A BS is a static host that does an address translation and message forwarding from a static network to wireless devices and vice-versa [7]. The area size differs from one to another. There are some studies that have been done in query results retrieval [3,4,9,13,16]. However, only a few have been done in the LDIS [1,2,4,15].

One study in [9] discussed the issues of how to retrieve query results within one BS. It divides a query scope into four regions and searches only certain regions based on the users movement. However, it does not deal with multi-BS. In a daily life, most user locations are mobile and change from one area to another. Therefore, a user movement may involve many BSs. When more than one BS is involved, delays might occur during this period, which results in handovers or others delays (such as transmission or processing). Handover is a process to transfer an ongoing call from one cell to another as a user

changes the coverage area of a cellular system [1,11]. Some approaches have been proposed in handling handover in [1,11,16].

In this paper, we propose an algorithm to retrieve query results in multi cells. The aim is to retrieve query results from multi cells accurately and to examine some factors that affect the retrieval time. We only concentrate on cases where users are not interested in the targets that have been passed and users are moving. One example of this case is found in a tourism context [5]. Normally, tourists have a tight schedule to visit some places in certain areas. If they have passed some places, they would not come back to visit those places. The term 'target' used in this paper is defined as any objects or places that are searched by the users.

We have some assumptions to simplify our discussion which are as follows: users are moving with steady velocities and directions; every BS has knowledge about its neighbours and the expected time to leave current BS; the predicted locations are known before receiving query results; and there are no errors in all partial data retrieved. Furthermore, the handover time is ignored since it does not make any change towards the prediction of users locations.

2. Related work

In this section, we review some existing studies on mobile query results retrieval. These include query results retrieval for LDIS applications, including how to retrieve query results from one Base Station and multiple Base Stations whilst a user is moving from one to another area [8–10,12,15,16].

The efficiency of query result retrieval in one BS for LDIS application was discussed in our recent papers [8,9] in which we define a scope, that represents a valid area for user queries. Then, we retrieved the query results from servers connected by one BS. To avoid unnecessary query results retrieval, we decide to retrieve the query results based on user movement. We assume that the users are travelling with constant speeds and directions.

Figure 1 illustrates our proposed algorithm to retrieve query results within a single BS. Let us consider, one user traveling to the east at speed 2 is sending a query, "retrieve all vending machines within 1 km from my current location". When the user accepts the query results, they must reflect all vending machines which are located 1 km away from the latest position, since the user is not on the sending location anymore. However, the user is only interested in the query results that have not been passed (shaded darker area). Therefore, the valid query results are the vending machines (V9, V10, V11, V13, V14).

In the work by Sistla et al. [12], the location of a moving object is conducted as dynamic attributes which are divided into three sub-attributes: *function*, *updatetime* and *value*. They discuss how the value of dynamic attributes changed over time is denoted by a function. Three types of queries: *instantaneous*, *continues* and *persistent*, are presented in their work. The first two types present the current and the future states in the database respectively. The series of instantaneous queries on the infinite history starting at certain time, are categorized as persistent query type. The contribution of their work is that a new prediction location can be found by using the function of time. Therefore, we adopt their method to calculate the prediction location in our approach.

Zheng et al. [16] categorize various handover mechanisms into four types: *Naive*, *Priority*, *Intelligent* and *Hybrid* mechanisms. The *Naive* method is the simplest of the four mechanisms to be implemented. However, the waiting time for answers from a server is shorter compared to the *Priority* method which can answer queries of normal users unless the number of urgent users keeps increasing. The *Hybrid* method does not give a better result because, if the number of users is large, the waiting time will be

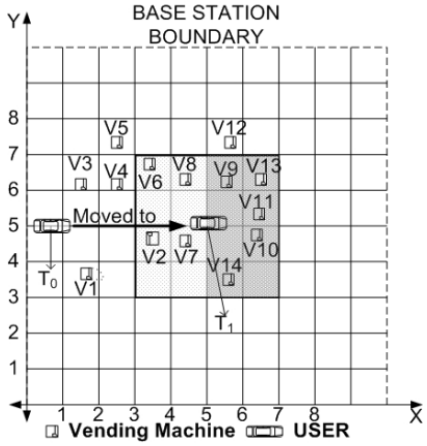


Fig. 1. Query Result Retrieval in a single BS.

lengthen. The *Intelligent* method gives a better result since this method does the calculation of the expected time to leave current cells. In this method, if the expected time to leave current cells is known; BSs of new cells know when they process the queries by assuming unexpected delays are not occurring.

On the other hand, Sykas et al. [11] propose another four handover approaches, namely *Ping-Pong avoidance* (*PPA*), *Towards the Border* (*TTB*), *MGIS Data Resolution* (*MDR*) and *Transmission Power* and *Interference Optimization* (*TPIO*). In the *PPA* approach, undesirable handoffs can be minimized by taking advantage the area information and mobility model to predict users movement. The *TTB* is useful for predicting when the users will reach the boundary of BS.

The *Intelligent* and *TTB* approaches have the same purpose which is to predict how long it takes the users to reach the BS boundary. However, the *Intelligent* approach is very straightforward in computing the reaching time in a new BS coverage. This approach ignores the movement direction. In contrast, the *TTB* approach considers user directions in computing the reaching time. Therefore, we adopt the *Intelligent* mechanism in our proposed approach.

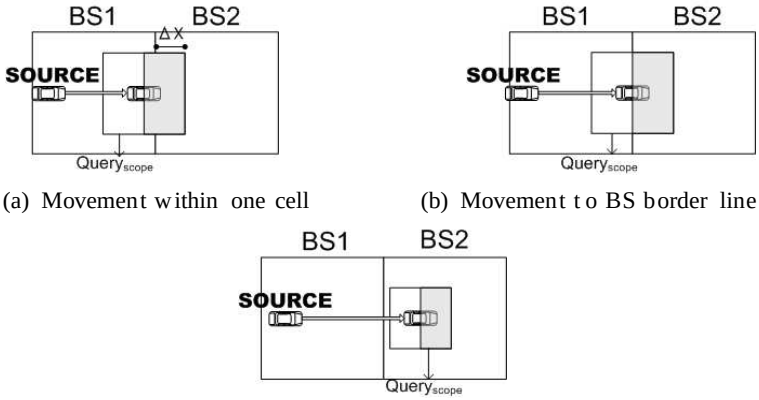
While our work is similar to theirs, our focus is to retrieve query results from multiple cells. The purpose of our work is to obtain query results accurately within a reasonable time.

3. Background

In retrieving query results, users can either stay in the same location or move to other locations. We are only interested in moving users. However, users travel with variable velocity and directions are not considered here. The aim of our proposed algorithm is to retrieve correct query results from servers where the query scope crosses the area of BSs.

While users are moving from one location to another, it is implied that the users are moving either within one or more cells. To calculate a user's new location, the function of time is used. The function of time multiplies a steady velocity by *time*. *time* is the duration from sending the query to receiving the results from servers. This duration depends on query processing from all participated servers.

The delay as a result of query processing can be either variable or fixed. A BS needs to have knowledge about the delays occurring in others. Even though the BS has the knowledge and knows how to predict the new locations of users, the delays occurring in one server vary. This situation can lead to an incorrect



(a) Movement within one cell (b) Movement to BS border line

(c) Movement within another cell

Fig. 2. Three types of users' movement.

prediction of location and give incorrect results to users. There must be a way to deal with these delays. However, in this paper we do not deal with variable delays. Dealing with a fixed delay time is not a big issue because the BS has to adjust the new prediction location by substituting the value of the delay time for the value of query result of the arrival time. Once this has been done, a new prediction location can be calculated.

After calculations of a user's new location, we classify their movement into three locations: within one cell, BS border line or new cell. If the users move within one cell or to BS border line (Fig. 2a and 2b), there is no handoff event. However, when the users move to another cell (Fig. 2c), handoff event occurs. Therefore, BS needs to predict when the users will leave its cell, and for this purpose we adopt the *Intelligent* method [16]. The *Intelligent* method is a method to process users' query based on computations when the users are expected to leave their current cells. This prediction is important for the future BS to maintain a queries queue of its users, since a user leaves the current BS area earlier than others.

We have stated that users move into three basic locations: within the same area, on the border line of two stations and next cell. When query scopes are performed, their boundaries can be classified into two types: crossing or not crossing BS boundaries. A query scope crosses if and only if it is crossing a BS's boundaries. Figures 2(a) and 2(b) show the examples of when a query scope crosses a boundary of two BSs. Otherwise, the query scope is said to be not crossing the BS's boundary (as shown in Fig. 2c).

4. Query results retrieval: Proposed Algorithm

In this section, we propose an algorithm for query results retrieval in multi-cells for location-dependent information services and provide some examples to justify our proposed algorithm.

4.1. Proposed Algorithm

In the previous section, we briefly introduced our proposed work as shown in Fig. 3. *Current BSID* represents the current BS identification number. BS_{scope} refers to endpoint coordinates that performed a boundary of BS. In our case, the number of endpoints used is four since we assume that the scope of BS is a square. All online BSs are stored in a collection, called *BS[1..n]* where *n* is an integer number

```
Algorithm: Query_Processing_for_Multi_BS
Input: Query Scope, Number of Neighbor Base Stations, Available Base Stations
Output: result
Begin
  Query_scope ← Scope of query
  BS_scope ← current BS station scope
  BS[1..n] ← online Base Stations
  Current_BSID ← ID of current base station
  Result ← Get_Result(current_BSID);
  While intersection (Query_scope, BS_scope) is true
     Previous_BSID ← Current_BSID
     Current_BSID ← next neighbour BS ID of the current BS
     BS_scope ← get_scope(Current_BSID)
     BS_scope.MIN ← get_scope_max(Previous_BSID)
     Query_scope ← Query_scope.MAX - BS_scope.MIN
     /* Append results retrieved to end of records */
     Result ← Result + Get_Result(current_BSID);
  End loop
  Return Result
End Query_Processing_for_Multi_BS
```

Fig. 3. The proposed algorithm.

to represent the number of online BSs. This collection contains a list of online BS which is sorted based on any BS registers. $Query_{scope}$ is four endpoint coordinates that represents the scope of user query.

After the parameters initialization, the current BS checks whether the scope of a query intersects the scope of the current BS. It generates query results in the current BS and stores the query results into a *result* parameter. If there is no intersection, only the query results in the current BS are returned.

If there is any neighbour BS next to the current BS, the current BS asks those BS, one by one, by passing the query and its boundary to those neighbour BSs. If there is an overlapping area between any boundaries of the current BS and its neighbour BS, the overlapping area is treated as the area of the current BS to avoid redundant processes.

While the current BS asks the user query of its neighbour, the current BS is treated as a user for its neighbour and the neighbour is treated as the current BS. The ID of the user (previous BS) is stored in the parameter *Previous BSID*. The scope of the current BS is retrieved and stored into parameter BS_{scope}. The minimum scope is replaced by the maximum scope of the previous BS. The minimum and maximum BS scope are defined as the closest and the furthest coordinates, respectively, from the new location of users based on the direction. Then, the query scope is deducted from the minimum endpoint coordinates stored in parameter BS_{scope}. The query results within the query scope are generated. These processes keep repeating until there is no intersection between the query scope and the BS scope. Then, the user receives the query results forwarded

4.2. Examples

In this section, we give some examples to show how our proposed algorithm worked. We divide this section into two parts: a non-overlapping BS area and an overlapping BS. First, we discuss how to retrieve query results based on examples in Fig. 2, where there is no overlapping area, after which, we discuss if there is any overlapping area.

As a running example, we use the following query "Find all ATM machines within 1 KM from my current location", which is sent to a server through BS1.

4.2.1. Non-overlapping BS area

In this section, we discuss the three illustrations in Fig. 2 where there is no overlapping area amongst BSs. Figure 2(a) shows a user moves into its neighbouring BS. Figure 2(b) when a user moves towards BS borders. Figure 2(c) shows the movement within the same BS and the query scope crossing the corresponding BS boundary.

As mentioned before, the illustration in Fig. 2(a) shows an user travelling within BS1 and the query scope is crossing the BS1 boundary. The interested query scope (shaded area) is decided by the user direction. When BS1 knows that the query scope is crossing its boundary, it processes the query within its area (shaded area from user location to ΔX) and gets partial about information of the query result from BS2 by forwarding the remaining query scope information from BS1 to BS2. Once BS2 finishes generating query results, it forwards the query results to its requester neighbour, BS1. Then, BS1 combines the partial query results retrieved from the other BS (Eq. BS2) and forwards the joined query results to the user.

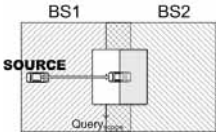
Figure 2(b) shows a user location at the border line of BS1 and BS2. In this situation, BS1 does not process the user query because, when the user misses the query results, BS1 needs to forward the user query twice to its neighbour. After BS1 receives the user query, it forwards the user query and the query scope to its neighbour, BS2, which processes the query and forwards the query results to the user immediately. In both figures, handovers do not happen since the user remains within one cell.

Figure 2(c) shows a user moving from his/her current BS1 to BS2. BS1 receives the query and calculates the prediction location of user. BS1 forwards the user query and the prediction of the user location to BS2 and the user query is handled by BS2. In this situation, generating query results of a number of users is quite difficult to maintain as we do not know when the users enter new cells. Knowing when users enter new cells is the easiest way to solve this problem. To achieve this solution, the *Intelligent* method [16] is adopted. Therefore, the next neighbour, BS2, knows when users enter its area. The remaining processes of query result retrieval are the same as in the previous example.

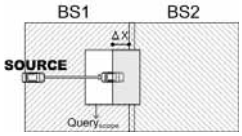
4.2.2. Overlapping BS area

Having presented some examples of non-overlapping BS, we now show some examples where the BSs area overlaps with others. There are many possible situations which can show a new location of users as the query scope interacts with BSs. We also provide examples where new locations of users are within the intersection area and before the intersection area.

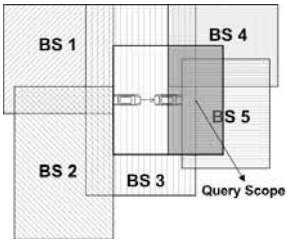
Figure 4(a) illustrates when a user moves to another BS and there is an overlapping area amongst the BSs. After BS1 receives the query from a user, it searches targets within its area. After that, it verifies whether there is an intersection with itself. If there is any intersection between BS1 and other BS neighbours, BS1 forwards the query to those intersected BSs. In this figure, BS1 forwards the query to BS2. BS2 adjusts its minimum boundary by assigning the maximum boundary of the BS1. Then, BS2 generates a new query scope by subtracting the current query scope from BS2 minimum boundary.



(a) Within an overlapping BS area



(b) Outside an overlapping BS area



(c) Within many overlapping BS areas

Fig. 4. Three Situations of Overlapping BS area.

Finally, all targets within the query scope and BS2 area are collected and sent them back to BS1. BS1 collects the results from BS2 and combines into its collection. The final results are sent to the user.

Figure 4(b) shows a situation in which a new location of users is before the overlapping area (shaded area between BS1 and BS2) and the query scope is beyond the BS1 area. In this situation, BS1 searches targets within its area and overlapping areas (shaded area). Once the searching process is completed, BS1 checks whether the query scope is beyond its boundary. If it is not, BS1 sends the query result to the requester.

On the other hand, if the query scope is beyond the BS1 area, BS1 forwards the query to all BSs which pass their areas. In this example, BS1 forwards the query to the BS2. BS2 searches its area by excluding the overlapping area since BS1 has already searched in that area. BS2 returns the query results to BS1 which returns the query to the requester.

Figure 4(c) shows one example of complex situations. Once the BS3 receives the user query, it seeks its area to match objects within its area with the user query. The overlapping areas of BS4 and BS5 are included too. Then, BS3 passes the user query to either BS4 or BS5, depending on which registers with BS3 first. If BS4 registers first, the overlapping BS5 is included. Otherwise, that area is excluded from the BS4 area. This situation is similar to that of BS5. If this BS registers first, the overlapping area BS4 is included in the BS5 area. Alternatively, the overlapping area does not belong in the BS5 area. After all areas where the user query passes return their answer to BS3, it joins those answers and sends them to the user.

5. Performance evaluation

This section presents the numerical results generated through a simulation. The purpose of simulation is to evaluate the proposed approach under various conditions. We describe our simulations, three categories of our testing, and the results of our simulation.

We have designed a simulation to test our proposed algorithm in Section 3 using Java™ programming Language. We run our simulation on two machines: 2.4 GHz AMD with 1 GB RAM and 700 MHz Pentium with 512 MB RAM. Both machines use Linux Fedora™ Core 1 Operating System. The simulation database contains various numbers of records of the random number *x*,*y*. In our experiment,

Table 1
First Parameters Setting

Parameter	Value
Number of BS	*5*
BS Area (M^2)	*250,000*
Query Scope (m^2)	*250,000–3,062,500*
Is every BS area different ?	*NO*
Number of User	*1–20*
User Coordinate (*x*,*y*)	(*400*,*400*)
Number of Item (every BS region)	*100,000–5,000,000*

Table 2
First Experiment Result

	Database Records			
Query Scopes (Metres)	100,000 records	500,000 records	1,000,000 records	5,000,000 records
100	5	28	54	265
250	29	169	299	1564
500	140	619	1260	6186
750	296	1380	2812	13929
1000	459	2190	4516	22336
1250	579	2864	5909	29300
1500	641	3176	6523	32469
1750	698	3485	7129	35619

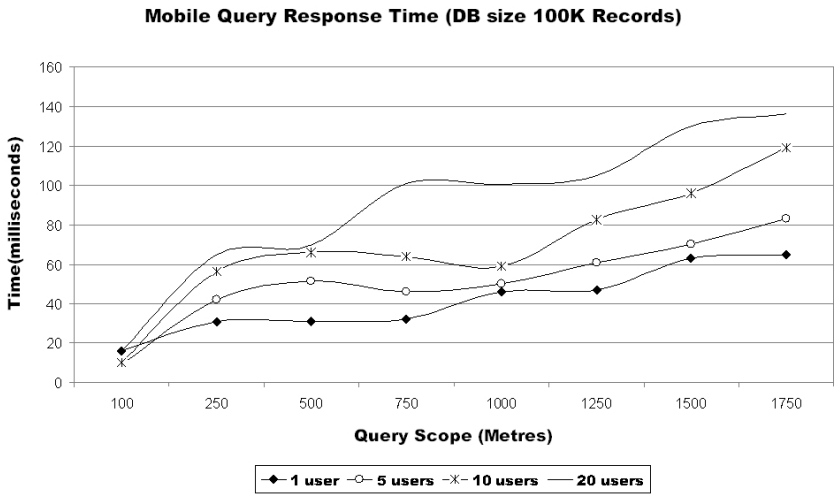
every BS is connected to a single database (DB) server, which contains from 100,000 to 5,000,000 records and the scope of a query is set to be beyond of the current BS boundary.

We have conducted three different types of analysis to examine our proposed approach. Firstly, we examine a situation where Multi-BS has the same area size and many queries with various sizes of query scopes. The purpose is to determine how long users should receive query results from a server. Secondly, we examine a stage where Multi-BS has different area sizes and variety queries with different sizes of scopes are sent. The purpose is to compare how long query results are received by users with either one or many BSs. In the last analysis, we follow up the second by examining the processing time for every BS if there are either a single or many users. From this experiment, we examine whether the BS processing time is responsible for producing query results.

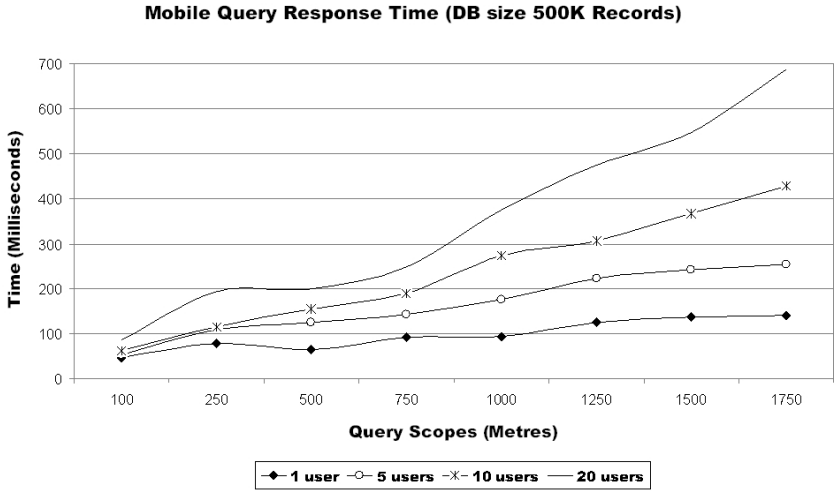
5.1. Uniform area size multi-BS and various size of query scopes

Here, we analyse the receiving time of query results from servers while users send queries with various size of query scopes from one location. All BSs have the same area size. The complete setting of our simulation for the first experiment is shown in Table 1. In this experiment, we used up to five BSs with the same area size. We tested 1 to 20 users sending queries concurrently. The query scopes are formed by squares and the areas vary from 250,000 upto 3,062,500 M^2.

Table 2 shows the results of the first experiment. We can see that users have more chances to get more targets within bigger query scopes. When the query scope is 100 meters, the users only get 5 targets if the total record of the database is 100,000 records. In contrast, users can get 698 targets when the query scope is 1750 meters within the same database. Furthermore, the other columns show that a smaller the scope of query shows has a smaller number of targets compared to a bigger scope.



(a) 100,000 DB Records



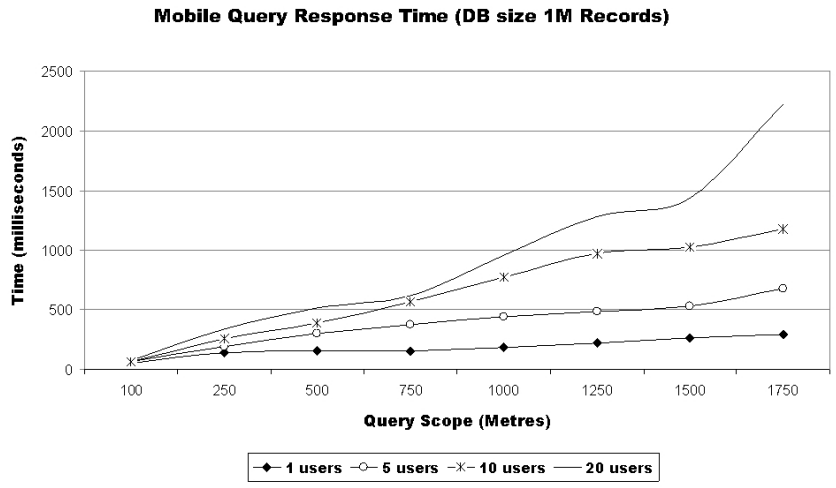
(b) 500,000 DB Records

Fig. 5. Various searching scope with 100,000 and 500,000 database records.

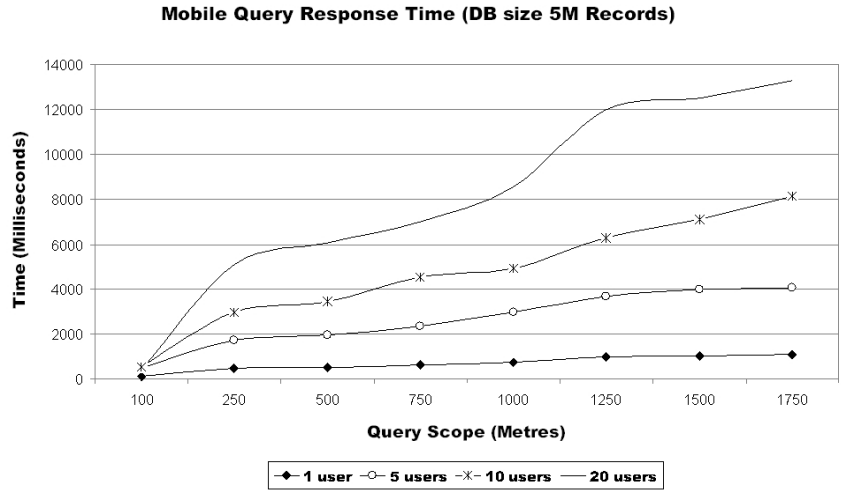
On the other hand, if we look the table horizontally, a larger number of database records shows more targets found compared to the smallest. From this point of view, we see that the number of targets found depends not only on the size of the query scopes, but also on the total records of database.

Figures 5 and 6 show the response time results where the database records are varied from 100,000 to 5,000,000 records respectively. Each graph shows the response time where the number users are: 1, 5, 10 and 20 users, sending the query until receiving the query results and the query scopes range from 100 up to 1750 meters. These graphs illustrate a general idea that a larger size of query scope and database records and more users increase the delays in answering user query.

Figure 5(a) shows the response time when the BSs accesses 100,000 database records. It shows that answering a smaller area of a query scope gives a faster response compared to a bigger area. In our



(a) 1,000,000 DB Records



(b) 5,000,000 DB Records

Fig. 6. Various searching scope with 1,000,000 and 5,000,000 database records.

simulation result, answering the largest area of query scope, 1750 metres, is slower by about 10 times than the smallest one, 100 metres. When we simulated for 20 users with query scopes of 750 and 1000 metres, our machine ran the daily updates. That is, the slope is slightly increased.

Figure 5(b) shows the response time when the BSs has 500,000 database records. It indicates a significant change while twenty users accesses the BS with a query scope greater than 1000 metres. The difference between the response time of the largest and the smallest query scope is similar to the previous graph. The current BS answers a user query of less than 100 milliseconds(ms) while a query scope is 100 metres. On the other hand, when the query scope is 1750 metres, the approximate time for the user to get the answer is from 100 to 700 ms depending on how many users request results. Furthermore, we can see that when the number of users processed doubles, the delay time is also doubled.

Table 3
Second Parameters Setting

Parameter	Value
Number of BS	*5*
Query Scopes (m^2)	*2,250,000*
Is every BS area different ?	*YES*
BS 1 Area (m^2)	810,000
BS 2 Area (m^2)	90,000
BS 3 Area (m^2)	250,000
Number of User	*1–20*
User Coordinate (*x*,*y*)	(*400*, *400*)
Number of Item (every BS region)	*100,000–5,000,000*

While the number of database records increases in size, the response time becomes slower. However, it is not only that; the query scope size is also responsible for responding to the user query. If we see Fig. 6(a), especially when there are more than 5 users accessing the BS concurrently, the increament of the graph is bigger than when there are less than 5 users. On the other hand, the response time to answer the targets within 1750 metres for 20 users significantly increases if the query scope is greater than 1500 metres.

Figure 6(b) is the last graph of our first simulation. It has same trends as the previous four. It shows that the response time is ten times slower than that shown in Fig. 5(a). It also shows clearly that if the number of users is doubled, the response time also increases by two. However, it is not guaranteed that the number of users is a primary factor in slowing down the BS response time. If we compare all four graphs, the number of database records is one of the factors which slows down the BS response time. This is due to the number of comparison required to be done in order to fullfil the criteria of user query.

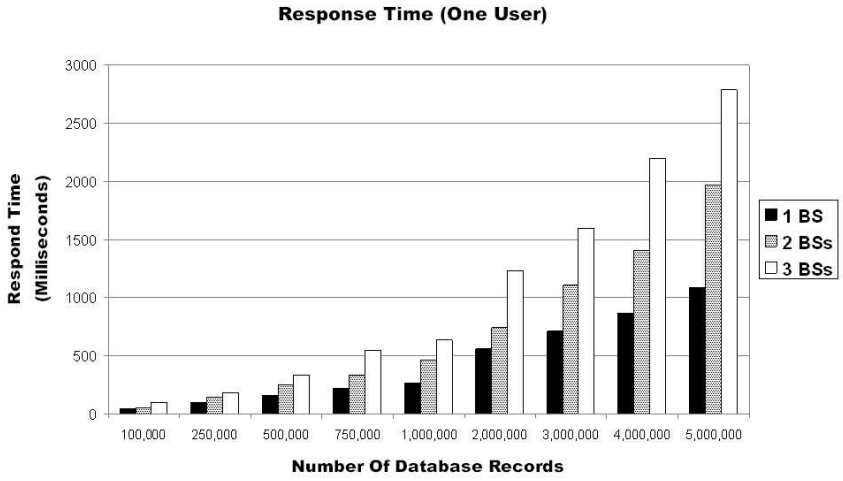
In summary, the longer that BS answers queries, the more targets are returned to the users. Furthermore, the size of query scopes and the number of records in the database also need to be considered. Finally, the total queries required to be processed concurrently also affect the performance evaluation.

5.2. Various area size multi-BS and uniform size query scopes

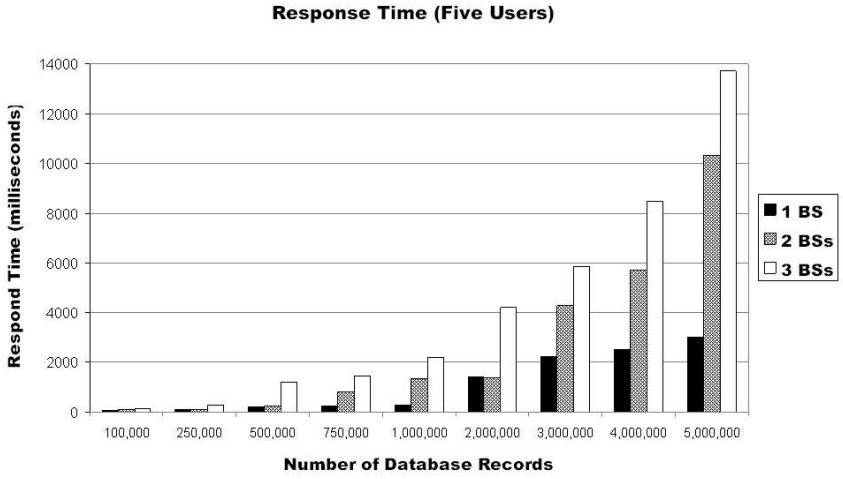
In the second experiment, many users send a number of queries, with the same query scopes, from the same position, to the corresponding BS. We use three BSs: one is the current BS and the others are the neighbouring BSs. Table 3 shows the parameter setting. In this analysis, we use three BSs where the area sizes of each BS are varied. However, all query scopes have same size. A range of total users send a query and the total records in the database are the same as in first experiment.

The purpose of this experiment is to measure the response time to answer user queries if there are many BSs and many queries involved at one time. An accurate result returned by BS is another point of interest. Table 4 shows the results returned to users if only a single or multi BS returns the query results. The response time of the experiment results displayed in Figs 7 and 8 are showing the response time if there are 1 to 3 BSs accessed by the number of clients, is 1, 5, 10 and 20 respectively. These figures show the same trends, that is, the more BSs involved, the longer time taken to give answers to users. In addition, when there are more queries processed at the same time, more time is taken to answer those queries.

When a user sends a query to one BS and the BS only searches within its area, the user will miss some targets or have to resend another query to collect targets within the neighbour. Resending another query to a new BS consumes more power and bandwidth. Table 4 shows the number of records found in the query result. The first and second columns show when only one or two BSs answer the query. Unless



(a) One User



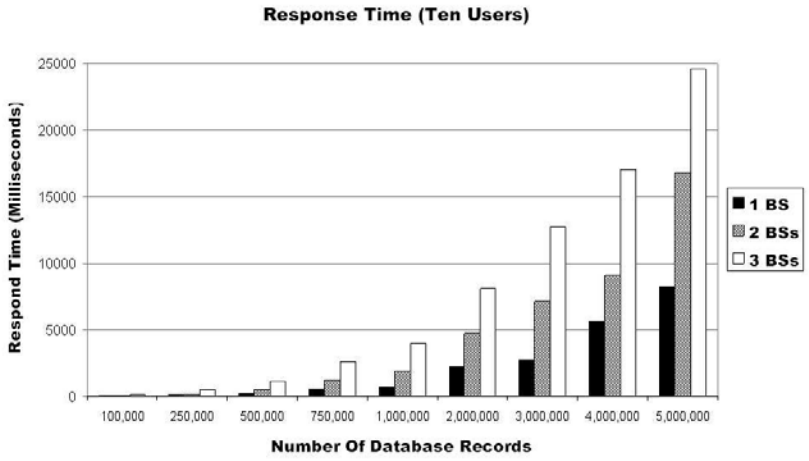
(b) Five Users

Fig. 7. A Single searching scope with one and five users.

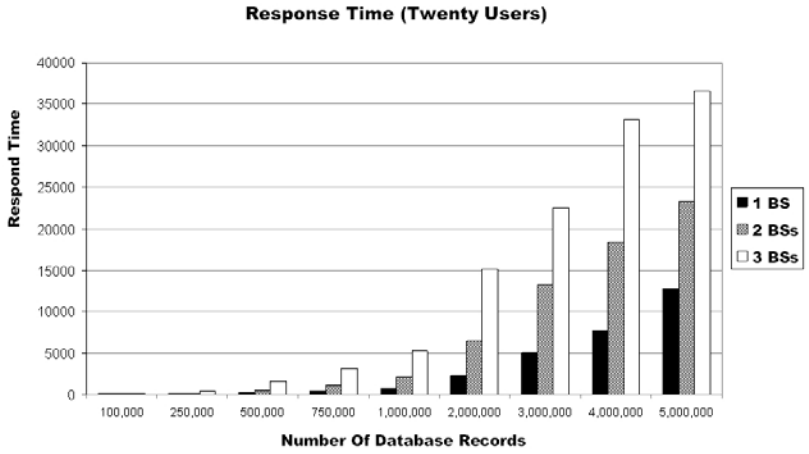
the third BS is down or failed to return a query result, the query result returned is insufficient since the user needs to resend another query when he reaches the neighbour BS.

Figure 7(a) shows the response time for giving solutions to a single user when there are 1 to 3 BSs used simultaneous. The database contains a number of records from 100,000 to 5,000,000. The average response time of accessing from 100,000 to 5,000,000 for three BSs, is around 32 percent compared to one BS only. On the other hand, the response time of accessing either 100,000 or 5,000,000 records is 61 percent slower than one BS.

Figure 7(b) shows the response time of five users which involves one, two and three BSs respectively. The delay time for responding to a request from users is up to 14 seconds for accessing three BSs with 5,000,000 records, while it takes 3 seconds for accessing one BS only. If we compare this graph to the



(a) Ten Users



(b) Twenty Users

Fig. 8. A Single searching scope with ten and twenty users.

previous one, it has a longer delay time. However, the average delay time for one user remains the same.

Figure 8(a) shows the results of the response time accessed by ten users for one, two or three BSs. In this situation, the delay in acknowledging request from a user is less than two times compared to previous figure. The BSs response to ten user queries is less 25 seconds for three BSs, 17 seconds for two BSs and 8 seconds for one BS. Others bars show same trend as two.

Figure 8(b) shows the response time of twenty users while accessing one, two, or three BSs. The response time of 5,000,000 database records is less than 40 seconds for three BSs, 23 seconds for two BSs, and 12 seconds for one BS. The trends of others bars are similar to the one discussed before.

The last four graphs above are similar from one to another. Therefore, we can conclude from this experiment that the delay in responding to user queries is n times slower to access n BS, where n is the number of BSs accessed simultenously.

Table 4
Second experiment result

Number of Database Records	1 Base Station	2 Base Stations	3 Base Statons
100,000	141	560	1,074
250,000	386	1,444	2,741
500,000	729	2,983	5,581
750,000	1,124	4,548	8,439
1,000,000	1,521	5,963	11,154
2,000,000	3,072	12,046	22,725
3,000,000	4,339	17,801	33,619
4,000,000	5,992	24,109	45,211
5,000,000	7,389	29,985	56,115

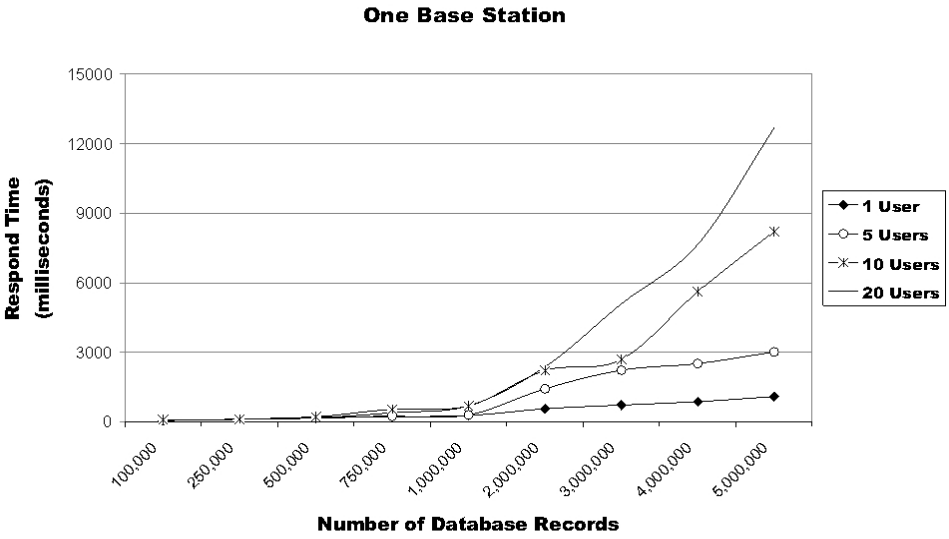


Fig. 9. Respond time of single BS.

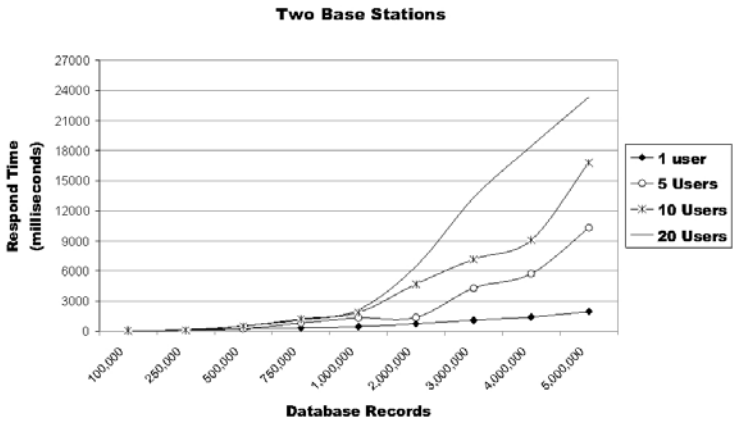
The next group shows the response time that is classified by the number of BSs. The purpose of this group is to shows when and why the respondse time of eachg BS is slowing down.

The next three figures, Figs 9 and 10, shows the response time for one, two and three BSs accessed by the same number of users with the same number of database records as mentioned in the previous group. The line is starting to slow while accessing 1 million records. However, they increase significantly if there are more than 3 million database records.

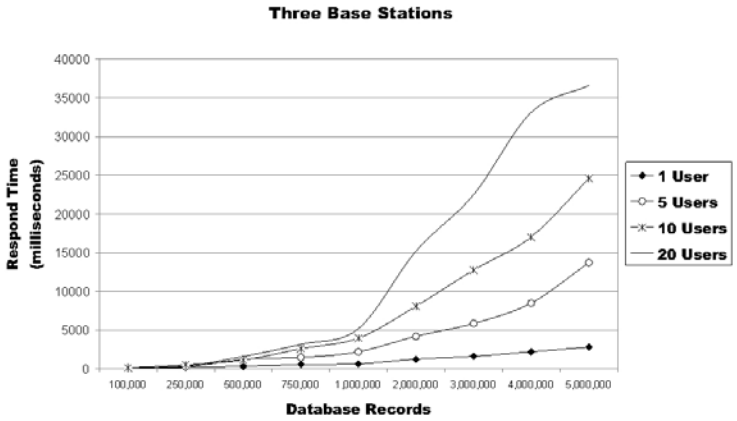
The increment of 10 and 20 users is significant compared to the last two lines. However, the average response time for each user is less than the response time of 1 user. The average is around 600 milliseconds.

Figure 9 shows the response time of single BS. It shows that the line increases slowly until it reaches 1 million records. After that, the response time slows down quickly. The response time of 10 users in a database contains 5 million records is a significant slow down.

After the response time of one BS, the next figure is the response time involving two BS as shown in Fig. 10(a). This increment starts higher after the database records reach 1 million records. The line that



(a) Two BSs



(b) Three BSs

Fig. 10. Respond time of multi-BSs.

presents the response time of 20 users is increased after 1 million records. However, if we calculate the average response time, it is still faster than the response time of 1 user only. On the other hand, the lines of 5 and 10 users show a parallel line. However, if we compare the gap between 10 and 20 users, the area is not twice as large as the gap between 5 and 10 users.

Figure 10(b) shows the response time of three BSs. The response time of three BSs is about twice as slow as those in last two graphs. This is due to data transmission and searching time for additional BSs. From this figure, we can see that the delay between two and three BSs is not much different when the database records less than 500,000 records. However, it starts slower if it accesses database records of more than 500,000.

5.3. Individual processing time of Multi-BS

In the last experiment, we measure the individual processing of each BS. We used the same setting as in the second experiment. The aim is to discover whether the processing time of each BS is reasonable.

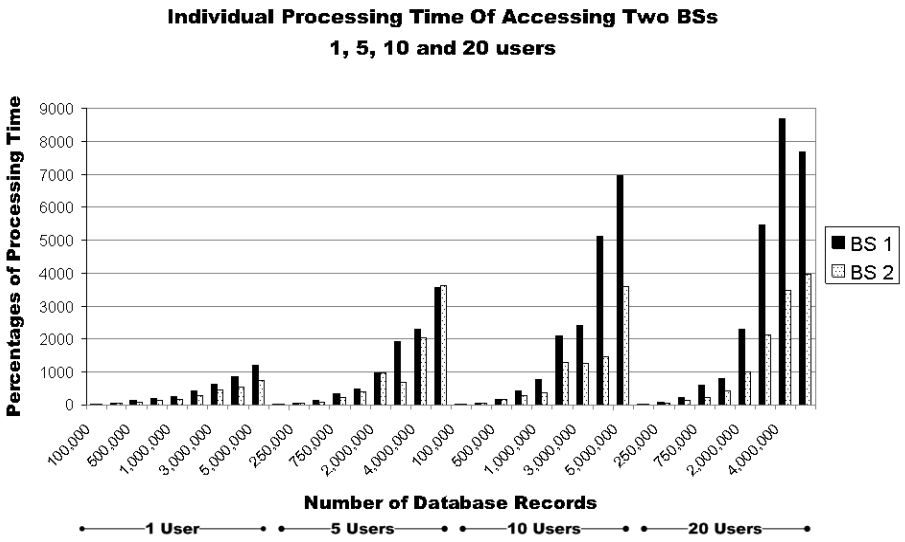


Fig. 11. Processing Time of Individual BSs for Same Query Scope and Two BSs.

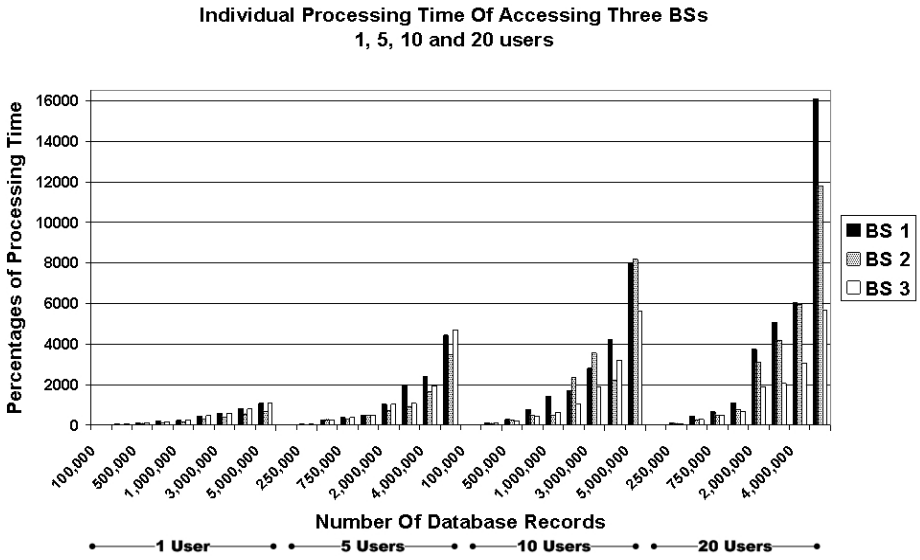


Fig. 12. Processing Time of Individual BSs for Same Query Scope and Three BSs.

We did this experiment twice. The first experiment used two BSs. The last experiment used three BSs.

The graphs shown in Figs 11 and 12 are the results of our experiments showing the processing time of each BS. A number different of users, from 1 to 20, are examined in this experiment.

When there are many BSs involved, we cannot say that the processing time of one BS (neighbour) are faster than another. This is due to the size of the database records and how rare are the targets matched with a query from the user. It is shown in the second group in Fig. 11 where the second BS needs more time to finish its process.

Figure 12 shows this issue clearly. In the first and second group, BS3 takes longer to finish its process than the other two. However, BS2 is the fastest to finish the process. In contrast, the fastest BS in the last group is BS3, whereas BS1 is the slowest to finish its process. BS2 is the last BS to finish its process whereas BS3 is the fastest BS.

6. Conclusion and future work

In this paper, we have shown how to retrieve query results in a multi-cell environment. In early sections, we explain our motivation on query results retrieval in a multi-cell. Then, we propose our algorithm and provide our analyses. We assume that the user travels on steady speed and direction. When a user moves from one cell to another cell, an unexpected disconnection, called handover, occurrs. Avoidance of lengthy handover time is done by calculating the expected of leaving time of the current cell.

Whenever there is any intersection between query scope and BS scope, it implies that the query results are within multiple cells. Therefore, the query scope, beyond the corresponding BS boundary, is subtracted from the minimum boundary of the next BS. This is due to the process of the remaining query scope inside the next BS.

In addition, we deal with delay time if any. However, we do not consider the variable value of delay time. When there is any fixed delay time, it is added to the value of the retrieval time. Our experiments results show accurate value. In addition, they show a reasonable value of time to process and respond to user queries.

We have shown that our proposed approach is effective when the number of database records is l million. The response time starts to slow down when the database contains more than 1 million records. On the other hand, the results show an accurate number of targets from the database matched with the queries requested.

On the other hand, targets move from one location to another with various speeds and directions. This situation will affect a result generated, therefore, we would like extend our algorithm as our future work to handle this situation.

References

[1] C. Abondo and S. Pierre, Dynamic location and forwarding pointers for mobility management, *Mobile Information Systems* **1** (2005).
[2] Z. Baihua, D. Lee, J. Xu and W. Lee, Data management in location-dependent information services, *IEEE Pervasive Computing* **1** (2002), 65–72.
[3] R. Benetis, J.C.S.G. Karciauskas and S. Altenis, *Nearest neighbor and reverse nearest neighbor queries for moving objects*, International Database Engineering and Applications Symposium, 2002, pp. 44–53.
[4] G. Cao, *A scalable low-latency cache invalidation strategy for mobile environments*, in Proceedings of the sixth annual International Conference on Mobile Computing and Networking, ACM Press, New York, USA, August, 2000, pp. 200–209.
[5] K. Cheverst, N. Davies, K. Mitchell and F.A., *Experiences of developing and deploying a context-aware tourist guide*, Proceedings of the sixth annual International Conference on Mobile Computing and Networking, 2000, pp. 20–31.
[6] M. Dunham and V. Kumar, *Using semantic caching to manage location dependent data in mobile computing*, Proceedings of the sixth annual International Conference on Mobile Computing and Networking, 2000, pp. 210–221.
[7] D.J. Goodman, *Wireless Personal Communications Systems*, Addison-Wesley Wireless Communications Series, 1998.
[8] J. Jayaputera and D. Taniar, Defining scope of query for location-dependent information services, *International Conference on Embedded and Ubiquitous Computing* **3207** (2004), 366–376.

[9] J. Jayaputera and D. Taniar, Query processing strategies for location-dependent information services, *International Journal of Business Data Communications and Networking* **1** (2005).
[10] A. Kahol, S. Khurana, S.K.S. Gupta and P.K. Srimani, *An efficient cache management scheme for mobile environment*, Proceedings of the The 20th International Conference on Distributed Computing Systems, 2000, pp. 530–537.
[11] A. Markopoulos, P. Pissaris, S. Kyriazakos and E. Sykas, *Efficient location-based hard handoff algorithms for cellular systems*, NETWORKING 2004,Third International IFIP-TC6 Networking Conference, 2004, pp. 476–489.
[12] P. Sistla, O. Wolfson and Y. Huang, Minimization of communication cost through caching in mobile environments, *IEEE Transactions on Parallel and Distributed Systems* **9** (1998), 378–389.
[13] I. Stanoi, D. Agrawal and A. Abbadi, *Reverse nearest neighbor queries for dynamic databases*, ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2000), 2000, pp. 44–53.
[14] X. Tang, J. Xu and D. Lee, Performance analysis of location dependent cache invalidation schemes for mobile environments, *IEEE Transactions on Knowledge and Data Engineering* **15** (2003), 474–488.
[15] A. Waluyo, B. Srinivasan and D. Taniar, Research on location-dependent queries in mobile databases, *International Journal on Computer Systems: Science and Engineering* (2005).
[16] B. Zheng and D. Lee, *Processing location-dependent queries in a multi-cell wireless enviroment*, Proceedings of the ACM international workshop on Data engineering for wireless and mobile access, 2001, 54–65.
[17] B. Zheng, J. Xu and D. Lee, Cache invalidation and replacement strategies for location-dependent data in mobile environments, *IEEE Transactions on Computers* **51** (2002), 1141–1153.

David Taniar holds a PhD degree in Computer Science, with a particular speciality in Databases. His research area has now expanded to Mobile Data Management. He has published more than 30 journal papers and more than 100 conference papers. He has published 6 books, including the forthcoming "Object-Oriented Oracle" (check the Amazon.com). Dr. Taniar is now with the School of Business Systems, Faculty of Information Technology, Monash University, Australia. He is an Editor-in-Chief of a number of international journals, and a Fellow of the Institute for Management Information Systems (FIMIS).

James Jayaputera is a PhD candidate in the School of Business Systems, Faculty of Information Technology, Monash University. His research interests are Mobile Computing and Data Management. He has published a number of papers in the area of mobile data management.