Multiresolution Amalgamation: Dynamic Spatial Data Cube Generation

Sham Prasher **Xiaofang Zhou**

School of Information Technology and Electrical Engineering
University of Queensland
St Lucia Qld 4072, Australia

{sham, zxf}@itee.uq.edu.au}

Abstract

Aggregating spatial objects is a necessary step in generating spatial data cubes to support roll-up/drill-down operations. Current approaches face performance bottleneck issues when attempting to dynamically aggregate geometries for a large set of spatial data. We observe that changing the resolution of a region is reflective of the fact that the precision of spatial data can be changed to certain extent without compromising its usefulness. Moreover most spatial datasets are stored at much higher resolutions than are necessary for some applications. The existing approaches, which aggregate objects at a base resolution, often results in a processing bottleneck due to extraneous I/O. In this paper, we develop a new aggregation methodology that can significantly reduce retrieval (I/O) costs and improve overall performance by utilising multiresolution data storage and retrieval techniques. Topological inconsistencies that may arise during resolution change, which are not handled by current amalgamation techniques, are identified. By factoring these issues into the amalgamation query processing, the retrieval loads can be further reduced with guaranteed topological correctness. Experimental results illustrate significant savings in data retrieval and overall processing time of dynamic aggregation.

Keywords: spatial aggregation, multiresolution, topology.

1 Introduction

Aggregating relational attributes has been widely studied, and is a fundamental operation in data mining and data warehousing, which are critical decision support tools in many applications (Chaudhuri, S. and Dayal, U. 1997). With the growth in geographic information systems, aggregation on spatial data, or amalgamation, is equally critical to the analysis of demographics over landmass or the distribution of rainfall and humidity in weather monitoring. The problem is that in these contexts, where many non-spatial attribute data are represented in continuous domains, user-driven classifications are unpredictable. For instance, when amalgamating regions based on common temperature ranges, user *A* can specify 30°-32° as 'hot', whereas user *B* specifies 30°-35°. Therefore the classifications used in an aggregation to merge all areas whose temperature is 'hot' are not known until run-time. A GIS may contain many such attributes that are used in combination to construct a spatial data cube. Hence predicting cubes for pre-generation is not possible.

While issues of selectively pre-materialising data to efficiently generate spatial data cubes are addressed before (Stefanovic, N., Han, J., Koperski, K. 2000), little work has been done on dynamic spatial aggregation. Amalgamation is used to re-classify objects into higher-level objects (HLOs) that form a new spatial layer. Thus the process is not simply visual, as investigated in previous systems (Ester, M., Kriegel, H.P. and Sander, J. 1997), but also involves geometry manipulation such that new geometries generated from amalgamation can be used in further spatial analysis.

We refer to an amalgamation that produces a large number of HLOs as *fine-grained*, and one that produces a few HLOs as *coarse-grained*. An efficient approach to coarse-grained amalgamations is presented in (Zhou, X., Truffet, D. and Han, J. 1999). The shortcoming of this approach is it has not been applied to fine-grain amalgamations. Also, it only considers data at a single resolution, or its original resolution (also known as level of detail, or LOD). This is inflexible when considering LOD is becoming more integrated in 2D/3D spatial query processing.

Multiresolution spatial databases are gaining popularity due to their versatility and efficiency in data retrieval (Jones, C.B. and Zhou, S. 2001, Horhammer, M. F. and Freeston, M. 1999). However, in the context of complex operations, they remain relatively unexploited. When attempting to extend additional spatial operations, such as amalgamation, we find current solutions do not adequately handle new topological issues that arise with changing the resolution of data.

Multiresolution amalgamation is the grouping of spatial features at varying resolutions of spatial object for two main reasons. Firstly, we may wish to relax constraints on the precision (resolution) of data, either because it is not necessary or to prioritise analysis speed over accuracy. This is particularly useful in data mining where we can efficiently generate approximate intermediate results that can be progressively refined if needed or found to be interesting, much like the filter-and-refine approach in spatial data processing. We may use low resolutions act

as an initial filter when performing amalgamation on datasets that are otherwise too large to process all at once. For example a nation-wide dataset could be processed at a low LOD to discover the general trend of HLOs. Interesting patterns, or 'hits', can then be isolated and refined individually using only data from specific areas of the spatial layer instead of the entire original dataset. Secondly, resolution can be changed to normalise the resolution of multiple heterogenous layers that will be processed together under some operations such as a map overlay. This is particularly pragmatic since comparing two geometries at different resolutions can only produce results at the precision of the lower resolution. Hence reducing LOD before processing would reduce the cost of the overlay.

In this paper we propose an alternative to the current approach to spatial aggregation, which treats the operation as an all-or-nothing atomic process, by applying multiresolution and topologically considerate techniques. In section 2 we outline previous work. Section 3 describes the multiresolution approach and experimental results are given in section 4. Conclusions are drawn in section 5.

2 Related Work

Modelling geospatial information as highly-customised views, comparable to those in relational systems, has opened new possibilities in areas of data retrieval and data analysis. The application of computational geometry to spatial data manipulation is not new (De Floriani, L., Puppo, E. and Magillo, P. 1999), though neither is it mature.

For purposes of visualization in OLTP, multi-resolution databases address the need to extract simplified spatial data from a sizable source to suit limited-capacity mobile devices, and to consider network transmission costs (Zhou, X., Prasher, S. and Kitsuregawa, M. 2002). In 3D applications hierarchical index structures are used to manage terrain meshes (Xu, K. 2003, De Floriani, L., Magillo, P. and Puppo, E. 1997). In 2D systems data objects are fragmented and sorted according to both location and relative map scale (Horhammer, M. F. and Freeston, M. 1999). The concept is developed further in (Jones, C.B. and Zhou, S. 2001, Prasher, S., Zhou, X. and Kitsuregawa, M. 2003) whereby objects are treated as a set of logically connected fragments. This proves beneficial since data retrieval is reduced through in-database object clipping and simplification, which mitigates both I/O costs in retrieval and the processing costs of subsequent external spatial operations.

Concurrent to spatial simplification, considerable work has addressed the preservation of topological relationships to ensure correctness of results in both 3D contexts (Gerstner, T. and Pajorola, R. 2000, Bajaj, C. and Schikore, D. 1997) and 2D contexts (Kuijpersm, B., Paredaens, J. and Van den Bussche, J. 1995, Ubeda, T. and Egenhofer, M. 1997). Methods generally accomplish this by either imposing topologically considerate data structures, or employing consistency checks when manipulating data. Tryfona and Egenhofer (Tryfona, N. and Egenhofer, M. 1997) discuss consistency in topological relationships between regions that may be aggregated, and of different resolutions (i.e. from heterogenous sources). This work focuses on topological inconsistencies that may result from aggregating disjoint objects into larger features not dissimilar to convex hulls. In contrast we focus on spatial layers that are divided into connected feature boundaries, and identify topological inconsistencies that arise from morphing objects as data resolution is lowered.

With respect to complex operations in spatial systems spatial join has received the majority of past research focus (Brinkhoff, T., Kriegel, H.P. and Seeger, B. 1993, Li, W., Gao, D. and Snodgrass, R. T. 2002) although results from joins and amalgamations can be used interactively for complex or large scale analyses. For instance data from a demographics layer is first amalgamated to a desired granularity then joined with another layer describing air pollution levels over the same geographic extent.

In (Ester, M., Kriegel, H.P. and Sander, J. 1997) database extensions were developed to support complex mining rules in the non-spatial properties of spatial objects with respect to their location – e.g. "real-estate value of land blocks with respect to distance to the city's centre". However the purpose of grouping is for visualization such that objects are cosmetically associated with each other through a visual cue - e.g. fill colour – not actual geometry. If used for further manipulation, such as joining or overlaying spatial layers, groups would be processed by treating each member object individually instead of using a single group geometry.

Procedural and algorithmic methods for amalgamation are well established in digital cartography (Peter, B. and Weibel, R. 1999). In (Zhou, X., Truffet, D. and Han, J. 1999) a single-resolution (SR) approach is used to effectively filter out extraneous data to the amalgamation, and reduce subsequent, external, processing costs. A quadtree index is expanded to incorporate information on the spatial occupancy of an object to help determine whether it is needed in the result. This structure is henceforward referred to as the *occupancy_index*. The dataspace *D* is decomposed into a set of peano cells. The area of the part of an object, *pid*, intersecting with a cell, also referred to as its *occupancy ratio*, is recorded. The index and data table schemas are given as:

```
occupancy_index(cell, pid, area)
data_SR (objectID, geo)
```

where *geo* stores the geometry of an object. If the total *occupancy ratio* of the objects in a cell is less than 100%, then the cell contains some part of the result because part of it touches unclassified empty space in *D*. Objects that intersect with this cell are considered boundary objects because they contribute to the final amalgamated boundary. Similarly the cells are termed boundary cells. The goal of the single-resolution amalgamation query (SAQ) is to identify boundary objects and filter out objects that do not contribute to the result. Once boundary cells are identified all boundary objects are retrieved and clipped to the boundary cells. Finally, line segments that

are common along boundary objects' borders are removed, forming the amalgamation result.

3 The Multiresolution Scheme – MR

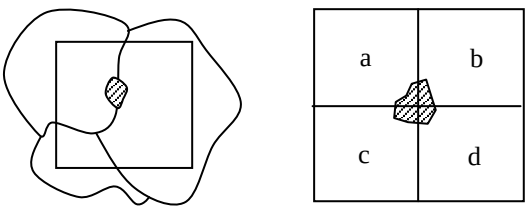
As with simplification, when resolution is altered topological issues arise when amalgamating geometries. To date no work has addressed, or defined, these issues. In this section we outline how the SR approach is extended to allow for varying grains of amalgamation. The goal of amalgamation is to identify all boundary objects and later merge them into HLOs in-memory.

A spatial layer D consists of a set of objects S, and unclassified empty space. This set has an equivalent collection of simple line segments L. Adjacent objects are amalgamated into HLOs if they share a common property t in a classifying attribute domain (CA) – e.g. temp=30-35°. We use CAt to denote the group of objects that share this property, and $F(S_t)$ as their amalgamated boundary, which has an equivalent set of unique lines L'_t where $L'_t \subset L$. Because objects in CAt can be disjoint ∂S_t can consist of many HLOs. For simplicity we refer to the generation of a single group boundary as $F(S) = L'$, and its corresponding set of boundary objects ∂S.

3.1 Topological Issues

In the single resolution scheme we can define a threshold below which any object is considered *small*. This is made under the assumption that amalgamation is performed on an original instance of the entire spatial layer being considered. Consequently, when a spatial filter (e.g. a query window) is imposed conditions change. The size of the region of interest is reduced, which reflects a change in the scale of the map. For instance, when using a layer of the Australia, a 5km^2 area may be considered as *small*. When a window is subsequently placed on the metropolitan area of Sydney this changes to a 400m^2 area. This dynamism also creates topological anomalies. We now elaborate on two such issues, and explain why they are problematic when using the *occupancy_index* structure to identify boundary cells.

The Lake Problem is concerned with removing *small* objects from consideration:



(a) (b)

Figure 1: Lake (a) and Jigsaw (b) problems

Figure 1 illustrates a cell containing three large regions surrounding a *small* area, known as a lake. To use a demographic example assume the lake represents a low-income city block, and the white regions are high-income areas. A single block contributes a minority of the total population within the entire area, thus may not be considered representative or important. By removing it we are able to provide a fuzzy classification of the whole area as high-income.

In the example, if the shaded region is removed, its neighbours merge, making the boundary cell an internal (non-boundary) cell. Therefore the participation relationship of an object with $F(S_t)$ is not static with respect to resolution. New internal cells should be identified and filtered out during query processing in order to minimise data retrieval. To accommodate this, occupancy levels of objects in cells must be adjusted according to the desired resolution at run-time.

Second, the Jigsaw Problem is concerned with distinguishing *small* objects from *small* object fragments. Objects that overlap multiple cells in the *occupancy_index* are partitioned into smaller fragments An identified lake could thus belong to a larger object. We term this the false lake anomaly. Removing part of a larger object can cause some topological inconsistencies in its simplified form when reconstructed into some HLO. For example if the part represents some significant geographic feature such as a peninsula.

An more pronounced situation is illustrated in figure 1b where an object is made up of numerous *small* parts that are individually removed, in turn discarding the object. This occurs because the current scheme for *occupancy_index* manages does not include any holistic object information, only data on parts.

3.2 An MR Object Structure

To represent an object O at some lower resolution r' a simplification method is employed. This produces a new object $O(r')$, which contains a subset of the points in O (i.e. $O(r') \subseteq O$. The object's original resolution is given as r, where $r' < r$. Performing simplification in-database requires that we obtain parts of a spatial object without fetching others from disk.

A storage scheme capable of $O(r')$ from O is structured as follows: An object is fragmented to a sequence of connected lines. The endpoints of a line segment are translated into to z-values using a quadtree decomposition of the data space (Orenstein, J. and Merret, T. H. 1984). Each line occupies a single tuple in the data table. During retrieval a resolution-sensitive spatial filter in processed on the line segments to produce $O(r')$.

3.3 Filtering on Resolution for Retrieval

The extent of D, *extD*, is often very large. To scale down an object's resolution a simplification method is needed. We employ the technique proposed in (Prasher, S., Zhou, X. and Kitsuregawa, M. 2003) because it performs simplification in query processing. Thus reducing overall data retrieval. Assume each point in D's space may be considered a display pixel p. When a low-resolution data space is superimposed and stretched to cover the area of a higher resolution data space its pixels are stretched. These elements are called *logical pixels*. The side (diameter) of a pixel in D is scaled accordingly (figure 2).

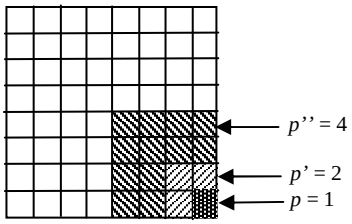


Figure 2: Logical pixels widths shown at two resolutions lower than original (*p=1*)

The change in *logical pixel* size roughly correlates to that of peano cells in data space decomposition. This reflects an intrinsic hierarchical property of z-values. By truncating the *n*-rightmost digits in a z-value the effective resolution of the resulting z-value prefix is reduced because it represents a larger area in *D*, which corresponds to a *logical pixel* in the new resolution. Because performing a truncation operation at run-time is costly we only note the position of *n*, at which a vertex and its succeeding neighbour share a longest common prefix. This position, called the *delta*, is computed between the starting and endpoint point of each two-point line segment in the dataset. At run-time, the difference between the data's original resolution, and the desired lower resolution is translated into a *delta* threshold. A large *delta* value denotes a long common prefix between endpoints, hence a fine detail. Lines having *deltas* lower than the threshold are added to $O(r')$. Thus simplification can be achieved using a single B^+-tree scan on line *deltas*. In terms of ensuring correctness of HLOs, this approach also has the advantage of preserving topological relationships, and avoiding issues of self-intersection and over-simplification.

3.4 Data Retrieval

During retrieval, the purpose of the amalgamation query is to detect boundary cells. This is done by first discarding *small* objects. To this end a minimal occupancy threshold (MOT), which is specified by the user, is used to as a filter. Moreover, because *small* is a relative measure, which is defined by query conditions (i.e. r'), *small* objects can only be detected at run-time, not pre-selected. However, if *occupancy ratios* are stored for each object fragment with respect to a particular cell, the MOT must be computed for each index entry because cell size is not fixed. This presents a problem for query processing as an index would then have to be constructed on an area-function for each possible r'. Clearly this is an unfavourable solution.

Another consideration is that we now wish to classify layers into many HLOs, as opposed to previous work which groups all objects into a single boundary. Thus we need to retrieve the CA information of objects intersecting with boundary cells. Consequently data is organised into three tables: a data table that contains information on object geometries, the *occupancy_index* table, and a secondary data table that records the CAs of objects. The latter two are given as:

```
data_MR(pid, lineid, z1, z2, delta)
occupancy_index (cell, PID, area)
object_table(objectID, CA1,…, CAn)
```

where the *z1* and *z2* fields record the endpoints of a line segment, and the *objectID* field is a foreign key to *pid*. Note that information in *object_table* can also be stored in *data_MR*, but would incur unwanted redundancy.

In previous work comparing the sum of areas with a cell's maximum capacity, 100%, identified boundary cells. We observe that if the layer's empty space is treated as an independent object, and recorded in *occupancy_index* with those cells it intersects, no cell will have a total occupancy less than 100%. This allows us to limit the query condition to searching for cells that contain non-*small* objects from at least two object groups. In this situation there exists some part of $F(S)$ in the cell. Because area is no longer needed to identify boundary cells we can instead record an object's area with respect to *D* in *occupancy_index*, and supply MOT as a static value. The multi-resolution amalgamation query (MAQ) is as follows:

```
SELECT Acell
FROM occupancy_index A, object_table B
WHERE A.pid = B.objectID
AND A.cell INSIDE window
AND A.area >= MOT
GROUP BY A.cell
HAVING COUNT(DISTINCT B.CA) > 1
```

Under the original scheme of *occupancy_index* computing the MOT is notably more complex. The MOT of *D* is given as a percentage, *T*. Occupancy is recorded with respect to a cell, whereas the MOT is a measure relative to *D*. The MOT is thus transformed into a percentage by being scaled to suit a particular cell *pc*. If *pc* is denoted by a z-value of length *f*, and *D* a z-value of length *g* where $g = \log_2 extD$, then the area of *pc* at *r* is $A(pc, r) = (2^{2(g-f)})$. In addition we must consider the decrease in resolution from *r* to r'. The area of *pc* at r' can be given as:

$$A(pc, r') = \frac{A(pc, i)}{4^{(r-r')}}$$

Finally, the MOT of *pc* at r' becomes:

$$\text{MOT} = \frac{T}{A(pc, r')}$$

Although the MOT can be calculated using standard built-in functions, it requires additional computation for each index entry, and information for each cell to be retrieved, i.e. the cell's z-value. This can be contradictory to the purpose of *occupancy_index* considering that larger indexes have proven to yield greater cost savings by providing more accurate filtering on cells (Zhou, X., Truffet, D. and Han, J. 1999). As with the MAQ we would also need to access CAs in the data table using a join.

3.5 Post-Retrieval Reconstruction

Objects that share the same value in the domain of a CA, and are geometrically touching, will be grouped to the same HLO. This is achieved by hashing the midpoints of

each line in ∂S then removing common lines (lines that share common midpoints) as well as belong to the same CA. At low resolutions objects are simplified by the removal of vertices, which can include the endpoints of common lines. The generic example in figure 3 illustrates two objects at their original resolution r connected by a common line denoted as line $[c, d]$ in object 1, and $[d,c]$ in object 2. At r' points d and e collapse, resulting in the selection of e, because they map to the same *logical pixel* (shown as the circle in figure 3). To ensure hashing is still applicable, and produces correct results, the midpoint of $[c,e]$ must be equal to that of $[c,d]$.

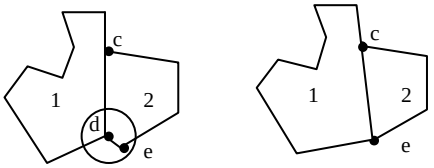


Figure 3: Collapse of common lines

Points are eliminated based on the simplification of one object. In object 1 d is retained whereas in 2 it is removed, changing $[d,e]$ to $[c,e]$. The coordinates of d and e at r are $\{dx, dy, ex, ey \mid dx \neq ex \wedge dy \neq ey\}$. At r' these become $\{dx', dy', ex', ey' \mid dx' = ex' \wedge dy' = ey'\}$. Hence $|cx\text{-}dx| = |cx\text{-}ex'|$ and $|cy\text{-}dy| = |cy\text{-}ey'|$.

3.5 Forming HLO Boundaries

Removing common boundary lines result in an unordered set of lines L'. To correctly re-construct object boundaries lines in L' are ordered as follows:

1. Both endpoints of lines in L' are hashed to a hash table. Each non-empty bucket is then processed, and references are created in an auxiliary array, A_{ref} for lines that share the same endpoint. This step collects the connectivity between lines of different objects that belong to the same HLO.
2. Lines in L' are then sorted on their object and line IDs. References in A_{ref} are shifted accordingly. Lines in L' are then hashed to buckets organised by object ID. Each bucket is subsequently sorted individually, ordering lines by object.
3. HLOs are reconstructed by following the linear order of lines in each bucket. Lines are processed sequentially until they reference a line from another object. In this case the scan jumps to the location of the referenced line and continues sequential scanning. When a closed HLO is formed it is stored in the result.

4 Performance Evaluation

In this section the performance of amalgamation is examined under the single-resolution scheme (SR) and multi-resolution scheme (MR). The aim of experiments is to answer the following questions:

1. How does resolution change affect performance?
2. Because the MAQ acts as an initial filter it should also be efficient and scalable. How does MAQ perform compared to the SAQ over coarse and fine grains of amalgamation?
3. What are cost benefits of making the MAQ sensitive to dynamic cell occupancy in order to solve the lake and jigsaw problems? Is it viable to instead use the SAQ in conjunction with in-database simplification , MR_Simp, to mitigate data loads and processing costs?

The amalgamation method should be applicable to different datasets. Figure 4 shows a growth in *small* objects, and associated cells, as resolution drops. One can deduce that if there is only a marginal change in the number of these objects from r to r', then the benefits of the MAQ are derived solely from simplification. In this case we could use the SAQ to identify boundary cells and the multi-resolution data structure to reduce data loads through simplification alone. However, the number of *small* objects is dependant on characteristics of the data, such as. distribution, shape and size of objects. This test is designed to illustrate that simplification alone is not sufficient to produce correct results.

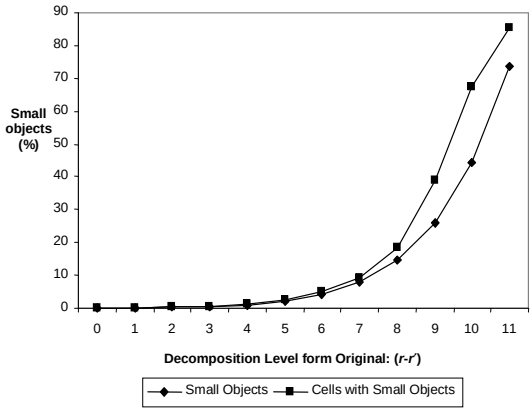


Figure 4: Number of *small* objects and cells containing *small* objects for each resolution below the original (0)

4.1 Data structures and Environment

Under SR objects are structures as in (Zhou, X., Truffet, D. and Han, J. 1999). A single data table contains the spatial geometries, corresponding object identifiers, and non-spatial attributes of each object. The *occupancy_index* records the *occupancy ratio* of a object within a cell. The tables used in SR are:

```
data_SR(objectID, geo, CA1, ..., CAn)
occupancy_index(pid, cell, area)
```

where *objectID* is a foreign key of *pid*. For MR we use the tables defined in section 3.4. Under both MR and SR, B^+-tree indexes are placed on the *pid* and *cell* fields. The TIGER/LINE[1] census block dataset for California is used. The dataset contains 21,648 objects in 1,618,950 vertices.

[1] http://www.census.gov/geo/www/tiger/

The *occupancy_index* contains 77,196 tuples. On average each cell contains 5.1 different objects, and each object is contained in 3.4 cells.

Under MR object and line identifiers, z-values, and the *delta* are stored as number types. We found a generally broad distribution of values in the *delta* domain, and used a composite B^+-Tree on the z-value and delta fields to exploit selectivity of the attributes. Under SR object geometries are stored as objects data types and indexed under an R^+-Tree, although tests were performed on the entire dataset with no window queries. Querying and processing on retrieved data was performed through a JDBC interface on a PIII800 256Mb RAM system.

4.2 Test Results

In this section we provide results of performing amalgamation under MR, SR, and MR_Simp, which is the combination of the SAQ with the in-database simplification used in MR. Amalgamation is performed in four stages for the MR and SR schemes:

Query: In SR Boundary cells are identified using the SAQ, and the IDs of all objects contained in those cells are retrieved. In MR all boundary cells are identified and retrieved into using the MAQ.

Fetch: In SR each object is fetched from disk then broken into a series of line segments. The object is clipped such that only lines that intersect with a boundary cell are retained. Under MR, line segments are retrieved directly from each boundary cell.

Merge: All line segments are hashed, and any duplicates are removed using algorithm SIMPLE in both schemes.

Reconstruct: Lines are reconnected into HLOs using the methodology outlined in section 4.5.

The dataset is encoded to a resolution of r=24, which represents point to approximately 2.8cm of accuracy. For resolution reduction we tested amalgamation of a 100% (i.e. the entire state) to levels r'=9 to r'=13. This gives levels of accuracy between 450m and 7.2km. For each resolution we performed two grains of amalgamation: a fine grain C1 that groups objects into 1806 (12 objects/group) HLOs, and a coarse grain C2 that groups objects into 58 HLOs (362 objects/group). Figure 5 illustrates performance for the first test: amalgamation on the entire dataset under each scheme for both coarse and fine grain settings.

In C1 (figure 5a) we test the *query*, *fetch*, and *merge* stages to illustrate that data reduction during querying is sufficient to cause significant savings in later stages, i.e merging. In C2 (figure 5b) all four stages are tested. A break-down of performance into the *fetch* and *merge* stages are shown in figure 6.
MR_Simp and MR perform comparably under C2, though a marked difference is shown when testing with larger data loads in the finer grain of C1 (figure 5a).

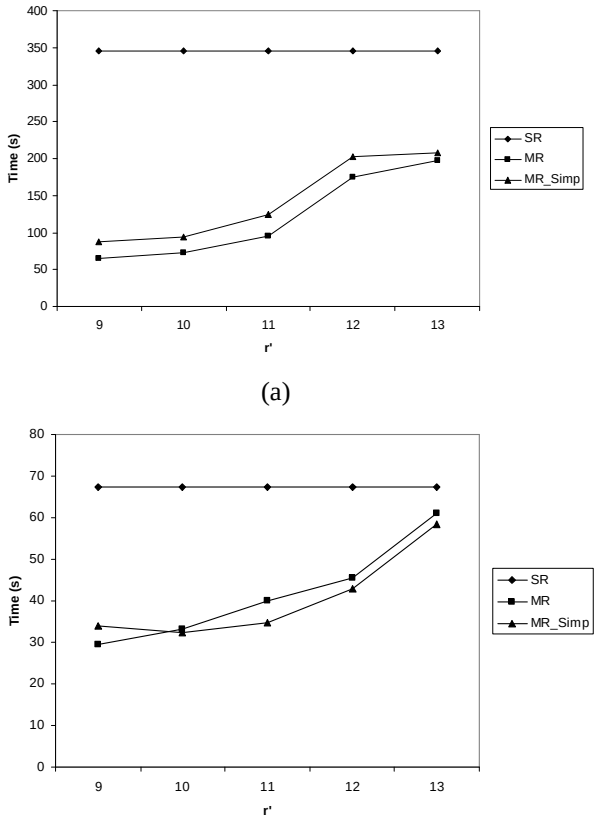


Figure 5: Performance of MR and SR variations in C1 (a), and C2 (b)

Figure 6a shows that MR_Simp generally retrieves more data than MR, which is then reflected in slower merge times in figure 6b. Figure 7 shows data loads retrieved under MR as a percentage of those retrieved under SR. Because trends for C1 and C2 are almost identical we can conclude that data reduction is achieved largely due to simplification. Therefore, because only a small part of performance in the first test are due to dynamic cell occupancy (i.e. boundary cells becoming internal cells after *small* objects are removed), we can conclude that MR can perform as well as MR_Simp in non-ideal situations; i.e. where cell occupancy is highly dynamic with resolution.

Data retrieval in MR (figures 6a and 6c) is clearly slower than SR, despite up to a 50% data reduction shown in figure7. We can attribute this behaviour to the high level of data fragmentation evident in the MR data table. Because line segments are stored separately, the additional storage overhead results in a higher overall I/O cost than the SR scheme.

Conversely, data reduction allows MR to perform merging significantly more efficiently than SR, as shown in figures 6b and 6d. We rely on hashing in-memory to improve merging by splitting a large set that must be sorted into numerous smaller sets.

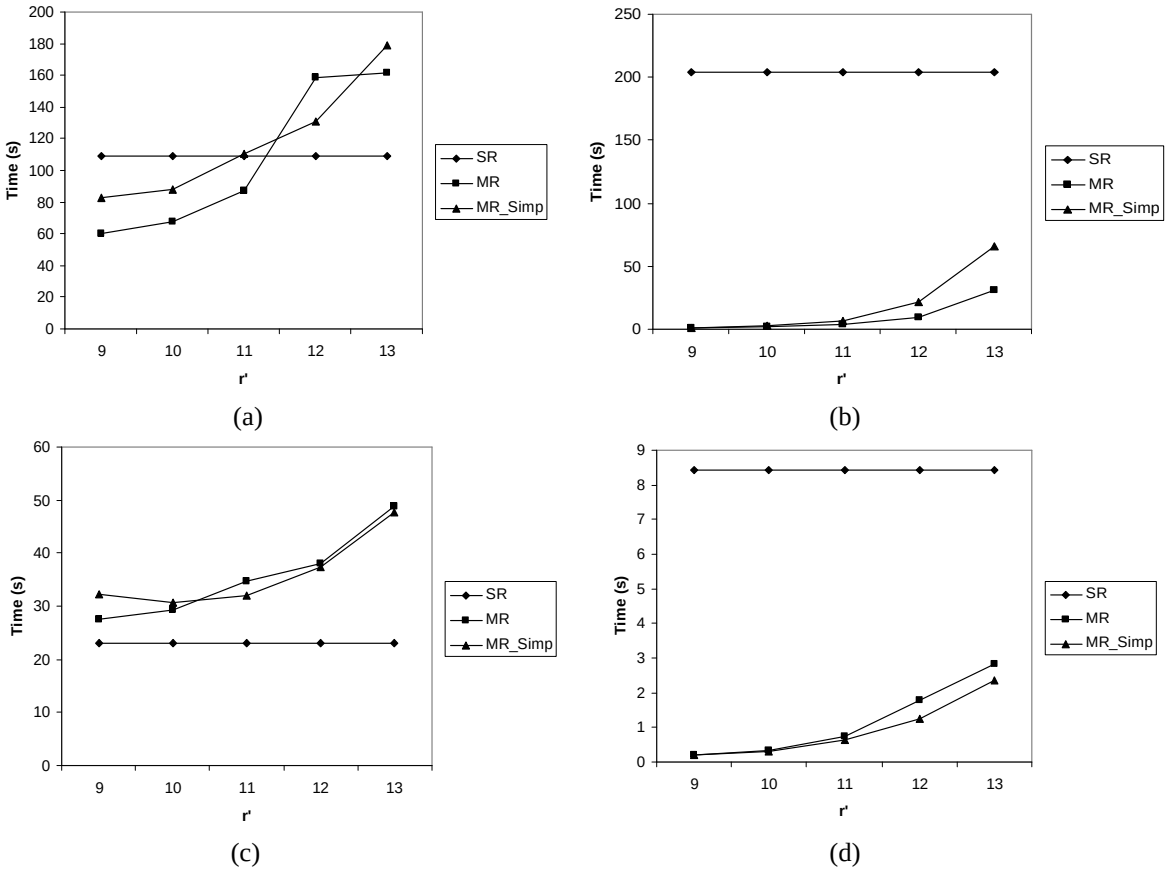


Figure 6: Retrieval and Merging under C1 (a) and (b) respectively, and C2 (c) and (d) respectively under SR, MR and MR_Simp.

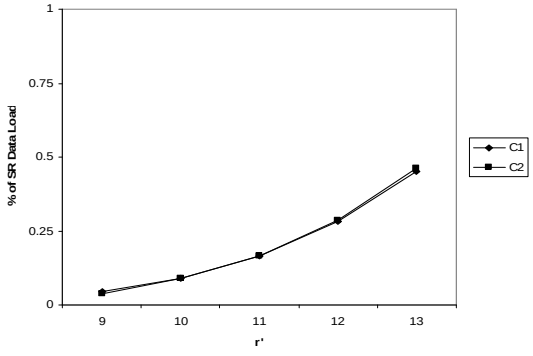


Figure 7: Data retrieval in C1 and C2 of MR as a percentage of SR

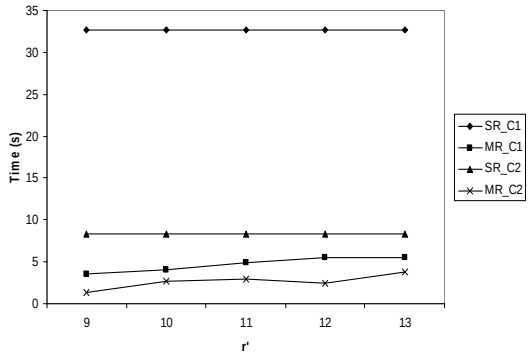


Figure 8: SR vs. MR query time

As SR continues to retrieve large data volumes, the sets stored in individual buckets of the hash table become large. Thus sorting on each bucket becomes ineffective, eventually contributing to a performance bottleneck. Clearly, the key to speed is tightly connected to maintaining some level of data reduction.

We now test the performance of the query stage under MR and SR. Figure 8 gives the results of performing both query types under the tests given in figure 6. The MAQ issues a range scan on the B^+-Tree to retrieve data for each cell. Hence it performs linearly with respect to the number of boundary cells identified.

The SAQ is required to retrieve objects and clip each object with each boundary cell it intersects. On average this step performs in average $O(nm)$ time where n is the number of objects and m is the average number of boundary cells per object (m <=3.4 in the tested dataset). This added complexity explains why SAQ scales poorly as the number of boundary objects increases with finer granularities of amalgamation.

In general a clear performance trend is observable between C1 and C2. Both are attributable to decrease in the data loads to be processed. As resolution approaches that of the original SR dominates MR because the

benefits conferred by detection of *small* objects and line simplification become minimal. Conversely when resolutions are set to lower levels MR significantly outperforms SR under both coarse and fine-grain queries.

In our final test, to determine the robustness of MR and MR_Simp, we test performance when cell occupancy plays a larger role in data reduction; i.e. when the lake problem is more prominent. In previous tests in C1 and C2 the number of cells that change between r and r'=13 is only 1.8% and 0.3% respectively. For C1 this gives MR_Simp approximately a 15% performance increase over MR_Sub. At r'=9 the difference in cells grows to 20%, but performance gained is only 3.5%. While a low change is clearly ideal, it cannot be guaranteed because cell distribution is dependant on both resolution and object classification that are given at run-time.

To simulate high cell changes a synthetic test layer is constructed: All *small* objects at r'=10 are allocated a separate classification that should be filtered out during query processing. Large objects are allocated to a different classification. Again reconstruction is omitted to illustrate how improvements to the merge phase through data reduction counteract higher retrieval costs. As expected, the results in figure 9 show a clear performance gain under MR because the MAQ considers *small* objects whereas MR_Simp retrieves extraneous, internal cells. As a result the amalgamated layer contains more HLOs that contain some unwanted noise; *small* objects.

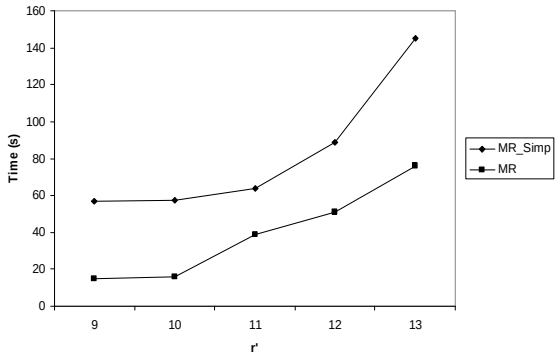


Figure 9: Amalgamation on synthetic dataset

In summary we observe significant time performance improvements from r'=9 to r'=13 of 82%-43% in C1, 56%-10% in C2, and 74%-48% in the synthetic dataset.

The results show that MR performs amalgamation in better time, for low resolutions, compared to SR. In contrast to MR_Simp and SR, MR proves robust when dealing with high dynamism in the classification of internal and boundary cells over various resolutions. When we consider this factor with object simplification, MR is clearly the most generalisable methodology.

5 Conclusions

In this paper a methodology to perform spatial aggregation on data at variable resolutions is presented. The main contribution of a multi-resolution approach is that the task can be performed at different levels of precision in order to prioritise processing speed. This provides two advantages comparing to the previous work: 1) Intermediate results can be quickly generated and examined before committing additional time and resources to a typically time-consuming amalgamation; 2) Use of low resolutions allows us to process datasets that are otherwise too large to process under current single-resolution methods.

We have also identified specific topological anomalies that arise during object simplification, which are not handled by the current single-resolution approach. The new multiresolution approach utilises in-database filtering during query processing to resolve two such problems. General performance under both coarse and fine-grain queries proved more efficient using the multiresolution approach due to effective data reduction. Additionally, by factoring topological changes into processing we are able to ensure the correctness of results to a level of precision given by the user-specified resolution.

Our contribution is important not only because of the growing need for multiresolution spatial support to accommodate different users or scales of analysis, but also the growing use of spatial warehousing and data mining in decision support systems. Our experimental results illustrated that an adequate data reduction leads to significant time savings. While pre-materialisation of spatial data cubes still performs ideally, they work only on a set of common views. Our tests imply that a greater scope of dynamic queries is more affordable using the new approach, making this operation practical for data mining and warehousing systems.

Acknowledgement

The work reported in this paper has been supported in part by grant DP0345710 from the Australian Research Council.

6 References

Bajaj, C. and Schikore, D. (1997): Topology Preserving Data Simplification with Error Bounds. *Journal on Computers and Graphics*, 22.1:3-12.

Brinkhoff, T., Kriegel, H.P. and Seeger, B. (1993): Efficient processing of spatial joins using R-trees. *ACM SIGMOD International Conference on Management of Data*, Washington DC, USA, 237-246, ACM Press.

Chaudhuri, S. and Dayal, U. (1997): An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65-74.

De Floriani, L., Magillo, P. and Puppo, E. (1997): VARIANT - Processing and Visualizing Terrains at Variable Resolution. *ACM-GIS*. 15-19.

De Floriani, L., Magillo, P. and Puppo, E. (1999): Data Structures for Simplicial Multi-complexes. *SSD*. 33-51.

De Floriani, L., Puppo, E. and Magillo, P. (1999): Applications of Computational Geometry to Geographic Information Systems, *In Handbook of Computational Geometry*. 333-388. SACK, R. AND URRUTIA, J. (eds). Elsevier Science.

Ester, M., Kriegel, H.P. and Sander, J. (1997): Spatial data mining: A database approach. *Fifth Symposium on Large Spatial Databases.*

Gerstner, T. and Pajorola, R. (2000): Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction. *Proceedings of the conference on Visualization*. 259-266.

Horhammer, M. F. and Freeston, M. (1999): Spatial indexing with a scale dimension. *SSD*, Hong Kong, 52-71.

Jones, C.B. and Zhou, S. (2001): Design and Implementation of Multi-scale Databases. *SSTD*. California, USA, 365-386

Kuijpers, B., Paredaens, J. and Van den Bussche, J. (1995): Lossless Representation of Topological Spatial Data. *SSD*, Maine, USA, 1-13.

Li, W. , Gao, D. and Snodgrass, R. T. (2002): Skew Handling Techniques in Sort-Merge Join, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Wisconsin, USA, 169-180, ACM Press

Orenstein, J. and Merret, T. H. (1984): A class of data structures for associative searching, *Proc. 3rd ACM SIGACT-SIGMOD Symp. on PODS,* Ontario, Canada, 181—190.

Peter, B. and Weibel, R. (1999): Using Vector and Raster-Based Techniques in Categorical Map Generalization. *Third ICA Workshop on Progress in Automated Map Generalization*.

Prasher, S., Zhou, X. and Kitsuregawa, M. (2003): Dynamic Multi-Resolution Spatial Object Derivation for Mobile and WWW Applications. *World Wide Web*, 6.3:305-325.

Stefanovic, N., Han, J., Koperski, K. (2000): Object-Based Selective Materialization for Efficient Implementation of Spatial Data Cubes. *IEEE Trans. On Knowledge and Data Engineering*, 12.6:938-958.

Tryfona, N. and Egenhofer, M. (1997): Consistency Among Parts and Aggregates: A Computational Model. *Transactions in GIS*, 1.3:189-206.

Ubeda, T. and Egenhofer, M. (1997): Topological Error Correcting in GIS. *SSD*, Berlin, Germany, 283-297.

Xu, K. (2003): Database Support For Multiresolution Terrain Visualization, *ADC*, Adelaide, Australia, 153-160.

Zhou, X., Prasher, S. and Kitsuregawa, M. (2002): Database Support for Spatial Generalisation for WWW and Mobile Applications. *WISE*, Singapore, 239-246.

Zhou, X., Truffet, D. and Han, J. (1999): Efficient Object Amalgamation Methods for Spatial OLAP and Spatial Data Mining. *SSD*, Hong Kong, 167 – 187.