VICTORIA UNIVERSITY OF WELLINGTON

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Efficient Retrieval of Structured Spatial Information from a Large Database

Eric K. Jones
Aaron Roydhouse

Technical Report CS-TR-94/20
December 14, 1994

Abstract

We show how a relational database can be used to store a large volume of complex, structured spatial information so as to support efficient retrieval by partial matching to a user's queries. A novel representation of spatial relationships is employed that facilitates efficient processing of conjunctive queries.

Our approach is implemented in MetVUW Workbench, a system for intelligent retrieval and display of historical meteorological data. The system is intended to serve as a "memory amplifier" for meteorologists that allows them to rapidly locate and analyse past cases of interest.

Publishing Information

This report also appeared in *Proceedings of the AI94 Workshop on Knowledge-Based Systems in Natural Resource Management*, Armidale, Australia, November 1994, pages 48–69. and has been submitted to the *International Journal of Applied Expert Systems*.

Author Information

Eric Jones is on the academic staff of the Department of Computer Science at Victoria University of Wellington, New Zealand. He received his Ph.D. in artificial intelligence from Yale University in 1992, and he also has degrees in mathematics and geography. His research interests include intelligent database retrieval, case-based reasoning, natural language processing, and machine learning.

Aaron Roydhouse is a graduate student in the Department of Computer Science at Victoria University of Wellington, New Zealand. He received a B.Sc. with Honours in Computer Science in 1992. He is currently investigating case-based methods for retrieving meteorological data.

1 Introduction

Effective exploitation of large archives of environmental data requires efficient identification of the data that are relevant for a particular task. As environmental data are spatially distributed, identifying relevant data is frequently a problem of spatial information retrieval. This paper addresses the problem of efficiently retrieving complex, structured spatial information from an archive of meteorological data.

The approach is implemented in MetVUW Workbench, a system for intelligent retrieval, analysis, and display of historical meteorological data (Jones and Roydhouse, 1994b; Jones and Roydhouse, 1994a). MetVUW Workbench is intended to serve as an intelligent assistant for weather forecasters and research workers in meteorology. Forecasters should be able to use MetVUW Workbench to quickly retrieve past cases that are similar to the current weather situation, and to analyse and display retrieved cases to facilitate forecast construction. For example, forecasters at the Meteorological Service of New Zealand currently engage in a "map session" every day at 10am, at which they compare alternative forecast hypotheses to arrive at a consensus forecast. MetVUW Workbench is designed to provide valuable input to discussions such as these. It is envisaged that forecasters will follow three steps when using the system:

1. Determine the meteorological processes and systems that appear to be driving the current weather situation.
2. Formulate a query to MetVUW Workbench in terms of these processes and systems.
3. Analyse and summarise the cases that are retrieved, or display any of the data associated with particular retrieved cases. Analyses and summaries can be used to provide evidence for or against the prognoses of numerical models, to help adjudicate between alternative forecasting hypotheses, and to draw attention to possible outcomes that forecasters may not have considered. Detailed examination of particular past cases can also usefully guide analysis of the current situation (Hunter, 1989).

Meteorology researchers should also be able to benefit from MetVUW Workbench, by using the system to rapidly identify large numbers of examples of empirical phenomena of interest. In the past, large-scale empirical investigations have been impeded by the difficulty of rapidly identifying relevant data. MetVUW Workbench addresses this problem by providing a convenient summary view of a much larger archive; this summary view is small enough to be stored on hard disk at comparatively modest cost. If more detailed analyses are required, larger amounts of data relating to a particular situation can be retrieved using slower media such as CD-ROM or magnetic tape.

MetVUW Workbench can be thought of as a kind of case-based reasoning system (Riesbeck and Schank, 1989; Kolodner, 1991; Kolodner, 1993). Each case is a slice of time for which meteorological data is available. The available data include satellite imagery stored both in digital form and on laser disc, a document archive, and numeric fields from the European Centre for Medium-range Weather Forecasting (ECMWF). Examples of numeric fields include pressure, temperature, relative humidity, wind speed, and relative vorticity, all of which are available at 12-hour intervals and 14 different levels of the atmosphere. MetVUW Workbench currently employs a portion of this numeric data that covers the Australasian region.

This paper focuses on case retrieval in MetVUW Workbench. Case retrieval proceeds by matching queries (sometimes called *index probes* (Domeshek, 1991)) against explicitly represented *index labels*. Queries identify particular features of the current situation that appear to be meteorologically significant. Index labels are representations of high-level features of the weather situation in a case, such as descriptions of high and low pressure systems. As far as possible, index labels are extracted automatically from the raw data. To date, the project has focused on features that are easy to derive automatically from ECMWF fields, such as the regions surrounding local minima and maxima.

MetVUW Workbench currently possesses 3.5 years of data, or a case base of some 2 500 cases. We anticipate that a further 10 years of data will soon be available to us, allowing the case base to be expanded to about 10 000 cases. Within several years, re-analyses of historical data by the ECMWF and other organisations should produce data sets covering a period from World War II to the present, which would permit construction of a database of over 36 000 past cases.

The large size of this case base gives rise to special difficulties that are not faced by many other knowledge-based systems. In particular, the set of index labels is too large to conveniently hold in memory. For example, index labels for low and high pressure systems over the Australasian region currently occupy about 65 Mb. As more high-level features of weather situations are added, and as the period of time covered by the database grows, storage requirements will further increase. We expect that a comprehensive collection of index labels for a 10-year data set covering the Australasian region will require on the order of 1 Gb of data.

Accordingly, MetVUW Workbench employs a relational database to store index labels. This approach is unusual in AI: most existing case-based systems have sufficiently small case bases that all indices can be stored in physical memory. The remainder of this paper is concerned with problems of intelligent retrieval of meteorological data using a relational database. Section 2 describes the architecture of MetVUW Workbench and indicates the role played by the relational database. Section 3 describes the representation of queries and index labels, and shows how index labels can be encoded using a relational database. Section 4 shows how the relational database can be used to select cases whose index labels are likely to match a user's query. Section 5 discusses the assessment of similarity between queries and cases selected from the database. Section 6 describes a way of augmenting the case representation to improve the efficiency of case selection. Section 7 summarises the key ideas of the paper.

2 The Architecture of MetVUW Workbench

Figure 1 depicts the architecture of MetVUW Workbench. The user interacts with the system via a *query constructor*, which allows queries to be formulated in terms of high-level descriptions of the meteorological systems and processes that appear to be relevant in the current situation. Users can describe high-level features of past weather situations, including low and high pressure systems, ridges, troughs, and jet streams. Eventually, it should also be possible to represent properties of these objects as they develop over time, such as the track of the pressure minimum of a low-pressure system.

Users construct queries by drawing representations of meteorological phenomena on a rectangular canvas that depicts a region of the globe. These phenomena are represented by points, lines, and regions, using a notation similar to an abstract weather map. This kind of interface was chosen so that the components of queries can be entered quickly and easily using a notation that is familiar to forecasters.

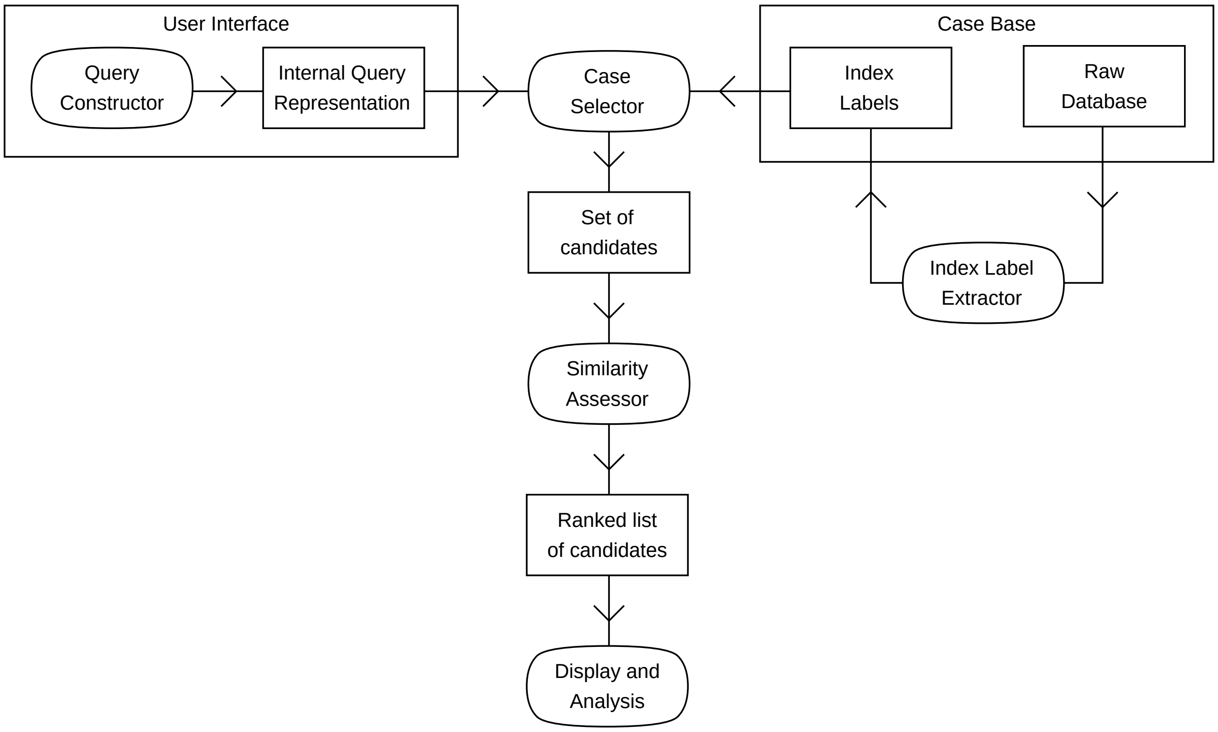


Figure 1: Architecture for case retrieval

A sample query is shown in figure 2. The query requests a complex low-pressure system over New Zealand, and two high pressure systems, one over Western Australia and another to the northeast of New Zealand. The position of pressure minima and maxima are marked, and the numbers indicate the pressure difference between the pressure at that point and the pressure at the perimeter of each system.

After a query is constructed, a symbolic representation of it is passed to the *case selector*, which uses key features of the query as indices for retrieving past situations or cases. The purpose of case selection is to ensure that the time required to retrieve a given number of cases scales sub-linearly with the size of the case base. (It follows that simply scanning the entire case base is an unacceptable approach.)

Case selection exploits the machinery of a relational database to efficiently access cases whose labels are likely to match a query. We use an advanced relational database called Postgres (Stonebraker and Rowe, 1986) to implement case selection. Case selection is efficient in part because it employs a number of sophisticated indexing strategies provided by Postgres, in particular, R-tree indexing of spatial information (Guttman, 1984). We will see, however, that to achieve maximum efficiency of case selection, it is useful to supplement Postgres' indexing mechanisms with an innovative representation of the spatial relations between components of index labels.

Once a collection of past cases has been retrieved, it is passed to the *similarity assessor*, which uses a knowledge-intensive partial matching process to rank the cases according to how well they match the query. Those cases whose match quality falls below some threshold are discarded.

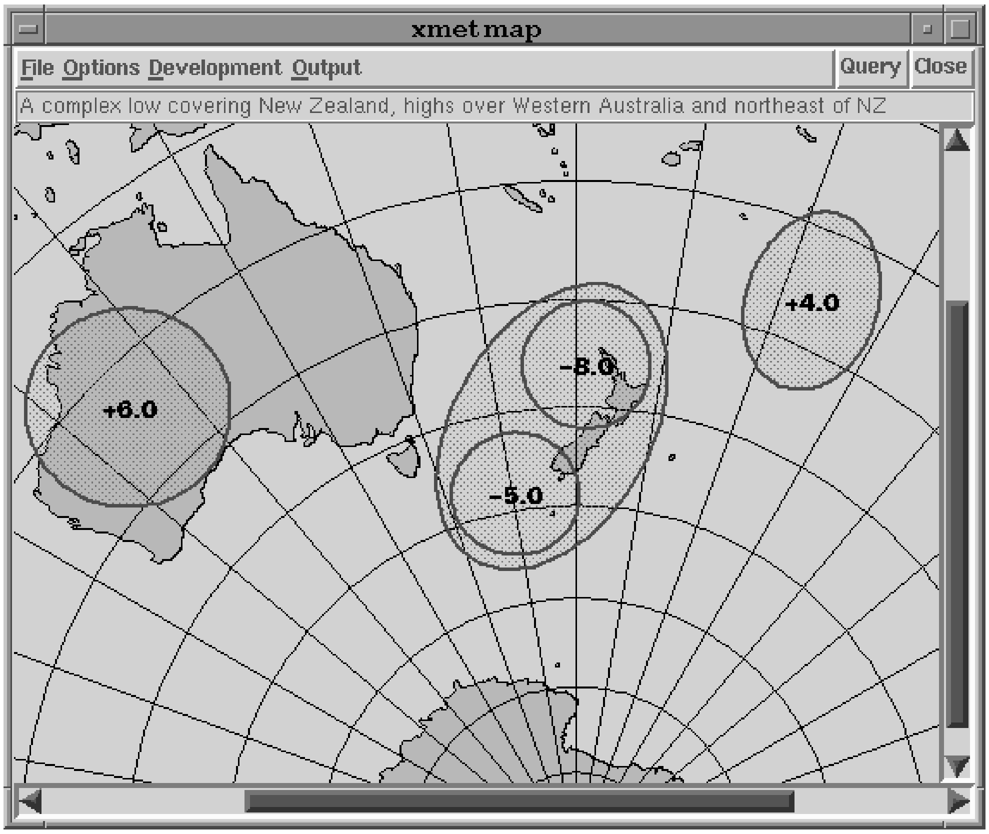


Figure 2: A query for a complex low over New Zealand, and highs over Western Australia and northeast of New Zealand.

Case retrieval thus proceeds in two stages. First, case selection uses Postgres' efficient retrieval machinery to identify a manageable number of candidate cases that need to be explicitly considered. Second, these cases are further filtered and ranked using a more costly but more accurate process of similarity assessment. This combination of case selection and similarity assessment facilitates fast, high-quality retrieval.

It is important to distinguish two notions of retrieval that arise in this architecture: case retrieval itself, and what we are calling case selection. Each has its own representation vocabulary for index labels. Case selection employs simple labels that meet the strict constraints of Postgres' retrieval mechanisms. In contrast, case retrieval employs a richer representation vocabulary that also supports similarity assessment.

For example, Postgres' R-tree indexing mechanism allows efficient retrieval of rectangular regions that are spatially related to a similarly oriented rectangular region in a query. A small set of built-in spatial relations is provided, including *contains*, *contained-in*, and *overlaps*. To exploit R-tree indexing, MetVUW Workbench employs rectangular bounding boxes of low and high-pressure systems as index labels for case selection. Moreover, MetVUW Workbench constructs Postgres queries that request cases containing systems that bear one of Postgres' built-in spatial relations to corresponding systems in the query. In contrast, similarity assessment uses arbitrary polygons to represent low and high pressure systems, and computes a more complex (and more informative) relation of graded match between systems in the query and the case.

3 Representing Queries and Index Labels

Queries are constructed using graphical objects such as points, vectors, and regions that denote high-level meteorological features. Corresponding to each graphical object in a query, there is an underlying symbolic representation that is used in case retrieval. Index labels are constructed using the same representation vocabulary. In this paper, we restrict our attention to high and low pressure systems.

In designing a suitable representation for high and low pressure systems, it is important to identify those aspects of their shape and relational structure that experts consider to have important meteorological implications. If case selection and similarity assessment are to produce useful results, these aspects must be explicit in the representation.

Individual regions of high and low pressure are represented in two different ways:

- A high-resolution polygon is stored that represents the outer boundary of the system. This representation is used during similarity assessment, to compute a detailed evaluation of similarity of shape.
- A rectangular bounding box to this polygon is also stored, oriented along lines of latitude and longitude. This representation is used during case selection.

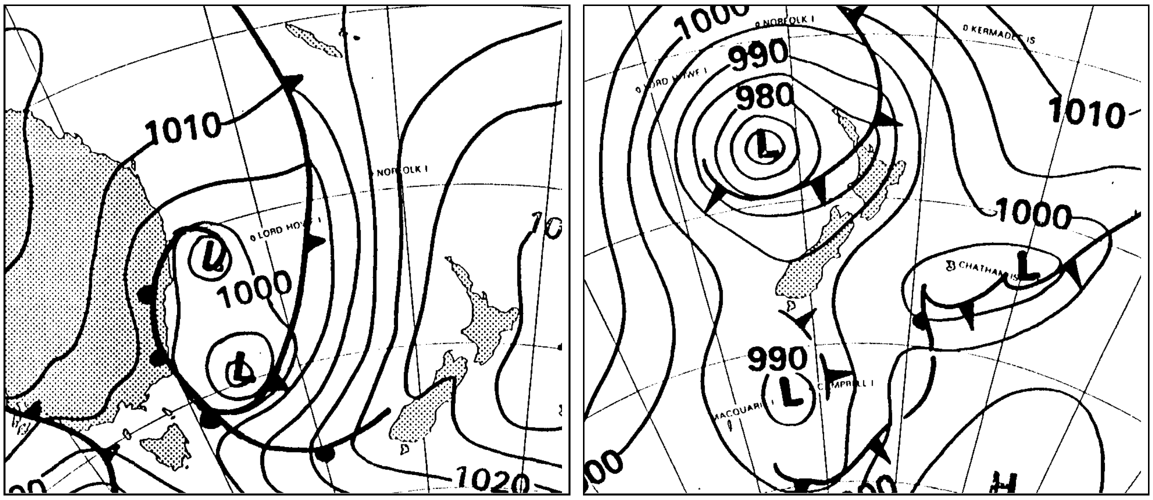
In addition, the ridges of high pressure systems are explicitly represented, because the presence or absence of a ridge tends to influence the weather at that location. This information is represented using a pinwheel of convergent axes, each axis of which denotes a ridge. Axes are labelled by their orientation and a measure of "curvature" or how well-defined the ridge is. (The representation of ridges is not discussed further in this paper.)

Low pressure systems often possess internal structure that is meteorologically significant: individual low-pressure regions are frequently clustered together into groups of two or more that overlap. Figure 3 illustrates two complex low-pressure systems. The entire region covered by such a system is likely to experience adverse weather, especially near centres of low pressure. It is therefore important to encode the shape of these systems in a way that makes clear the location of each subsidiary pressure minimum.

Complex low-pressure systems are represented as *clusters* of simple low-pressure regions. A cluster is a tree whose leaves are representations of simple low pressure systems—systems with a single low-pressure minimum—and each of whose internal nodes is a low-pressure region representing the outer boundary of a smaller cluster. A cluster is an ancestor of another cluster in the tree if and only if there is a pressure contour associated with the ancestor that completely encloses the contours of its descendant. This tree structure is used during similarity assessment to compute a measure of the degree of structural match between a query and a case.

3.1 Representing index labels using Postgres

All representations of index labels in MetVUW Workbench are automatically extracted from the ECMWF data and stored in a Postgres relational database. A portion of the index label for a typical case, describing low pressure regions, is given in table 1. Regions of high pressure are stored using a similar scheme. A graphical representation of all the low and high-pressure regions in this index label is depicted in figure 4. Each case is allocated a unique identifier, stored using the "Case Id" field.



Midnight, 4 August, 1990 NZST   Midnight, 17 August, 1990 NZST

Figure 3: Two complex low pressure systems

Reg. Id	Case Id	Parent Id	Pres. Centre	Min Pres.	Bounding Box	Peri. Pres.	Children
1798643	1798631	—	(205.8,-47.0)	998.0	(201.3,-57.2,211.9,-41.5)	1002.0	0
1798644	1798631	—	(157.9,-44.6)	994.0	(131.5,-54.9,184.1,-34.7)	998.0	2
1798645	1798631	1798644	(172.0,-43.8)	993.0	(156.5,-47.1,183.1,-36.3)	996.0	2
1798646	1798631	1798645	(178.9,-44.8)	993.0	(173.9,-46.3,182.6,-41.7)	995.0	0
1798647	1798631	1798645	(165.2,-42.8)	992.0	(158.3,-46.5,172.0,-37.2)	995.0	0
1798648	1798631	1798644	(143.8,-45.5)	994.0	(136.7,-52.9,148.2,-41.9)	996.0	0
1798649	1798631	—	(102.0,-50.9)	979.0	(81.7,-66.3,123.2,-36.4)	992.0	2
1798650	1798631	1798649	(100.9,-45.3)	979.0	(98.2,-47.2,103.6,-43.4)	979.0	0
1798651	1798631	1798649	(103.2,-56.6)	967.0	(91.1,-63.3,113.9,-49.5)	979.0	0
1798652	1798631	—	(168.5,-67.9)	984.0	(159.5,-71.3,177.7,-66.5)	986.0	0
1798653	1798631	—	(213.0,-75.2)	972.0	(183.0,-83.5,258.8,-68.8)	982.0	0

Table 1: Portion of an index label describing low-pressure regions, excluding the perimeter field.

Similarly, each region of low pressure has a unique identifier, stored using the “Reg. Id” field. The location of the pressure minimum for each region, and the mean sea-level pressure at this location are stored using the “Pres. Centre” and “Min Pres.” fields. The “Perimeter” field holds a high-resolution polygon representing the outer boundary of the region. (For brevity, this field is omitted from table 1, but it is shown graphically in figure 4.) A rectangular bounding box that approximates this polygon is also stored in the “Bounding Box” field. The “Peri. Pres.” field holds the pressure at the perimeter of the polygon.

As discussed above, complex low-pressure systems are represented as clusters of simple low-pressure regions. Clusters are built up out of individual regions of low pressure; the tree structure of a cluster is encoded using the “Parent Id” field. Regions of low pressure are used to represent both the simple low pressure systems at cluster leaves and the enclosing contours that make up the internal nodes of the tree. These contours are chosen to be as large as possible without sacrificing cluster structure. The “Children” field of a region contains the number of immediate subordinates in the cluster rooted at that region. The structure of a typical complex low-pressure system is depicted in figure 5. The labels on the tree match those in figure 4, and the numbers in parentheses match the Reg. Id’s in table 1.

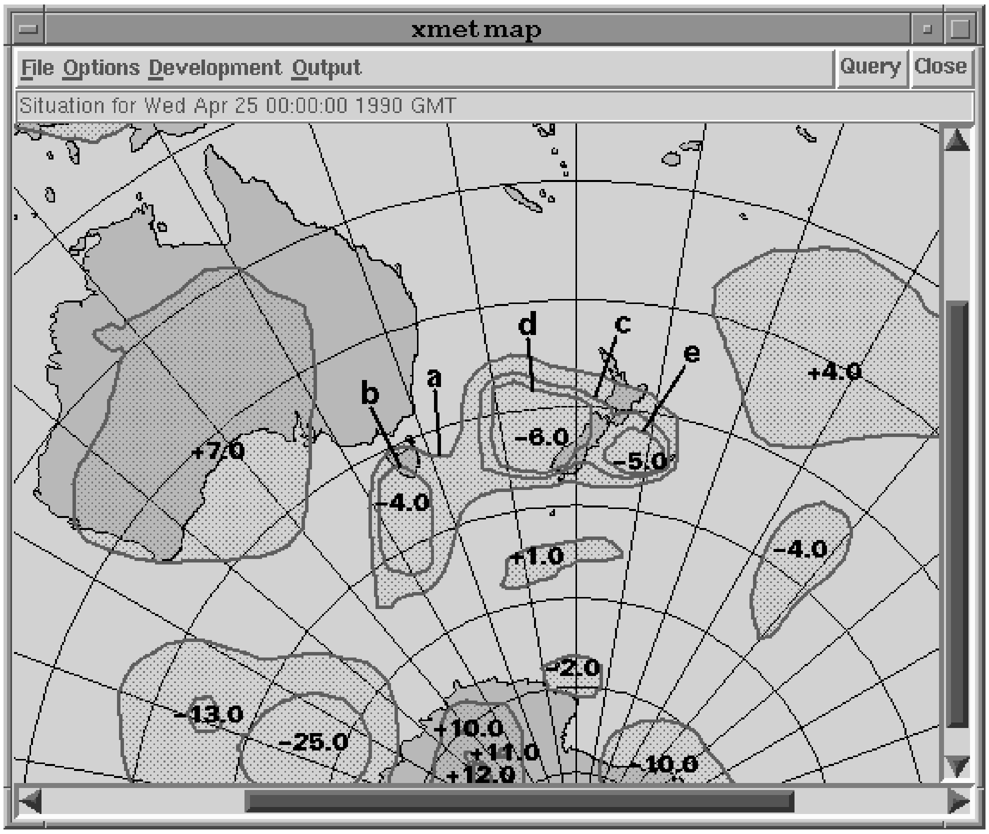


Figure 4: Graphical representation of an index label. The values indicate the difference between the pressure at that location and the pressure at the perimeter of the entire weather system.

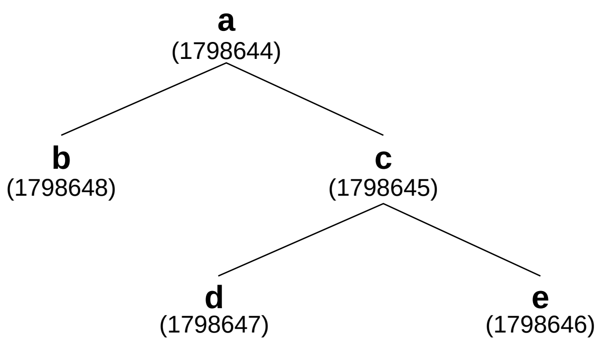


Figure 5: Structure of the complex low-pressure system over New Zealand.

4 Case Selection

Case selection proceeds in two stages. In stage 1 of case selection, Postgres queries are constructed and executed for each weather system in a user's query. The Postgres query for each high or low pressure system retrieves the all regions in the case base that satisfy the following requirements:

- The region has the same classification (high or low pressure) as the system in the query.
- The region's bounding box overlaps the bounding box of the system in the query.
- The pressure extremum of the region is contained in the area of overlap between the region and the system in the query.
- Similarly, the pressure extremum of the system in the query is contained in the area of overlap.
- If the system in the query contains subordinate regions, then the corresponding region in the case has at least as many immediate subordinates.

Each Postgres query is implemented as a *select/project* operation. For example, consider figure 2, above, which depicts a query from the user involving three weather systems. If these weather systems are denoted h_q^a, l_q, and h_q^b (moving from left to right), then the following three queries are issued to Postgres during stage 1 of case selection:

```
select into R1 (R.case-id) from R in REGION_HIGH where
  bounding-box(h_q^a) contains R.centre
  and R.bounding-box contains centre(h_q^a)

select into R2 (R.case-id) from R in REGION_LOW where
  bounding-box(l_q) contains R.centre
  and R.bounding-box contains centre(l_q)
  and R.children >= children(l_q)

select into R3 (R.case-id) from R in REGION_HIGH where
  bounding-box(h_q^b) contains R.centre
  and R.bounding-box contains centre(h_q^b)
```

Each Postgres query selects Case Id's for all regions (1) whose `Centre` (point of minimum or maximum pressure) falls within the rectangular bounding box of a particular weather system in the user's query, and (2) whose own rectangular bounding box overlaps the centre of the system in the query. Postgres employs R-tree indexing to efficiently retrieve these regions. For complex weather systems in the user's query such as l_q, only a single Postgres query is issued, corresponding to the outermost pressure region in the cluster used to represent the weather system. However, this query constrains the retrieved regions to be the roots of clusters with at least as many `children` (immediate subordinate regions) as the cluster in the user's query.

A certain degree of latitudinal and longitudinal displacement is permitted between systems in a user's query and corresponding systems in a case. A greater degree of longitudinal (east-west) displacement is permitted than latitudinal (north-south) displacement, because weather systems tend to naturally progress from west to east, and displacement in this dimension can often be attributed to infrequency of sampling. Latitudinal displacement is more significant, however, because the observed weather at a given location is considerably affected by latitudinal deviation, and because the physical behaviour of weather systems is strongly conditioned by their latitude.

Allowance for latitudinal and longitudinal displacement is encoded in Postgres queries by expanding the bounding box of each system in a query by fixed amounts in each dimension.

In stage 2 of case selection, the sets of Case Id's retrieved in stage 1 are intersected using a relational *join*, producing the set of cases that possess regions corresponding to every stage 1 query. For example, the following query to Postgres implements the join for the query from the user depicted in figure 2, above.

```
select (R1.case-id) where
  R1.case-id = R2.case-id
  and R2.case-id = R3.case-id
```

The output of this Postgres query is a list of Case Id's and is returned as the result of case selection. The case depicted in figure 4 is one of the cases selected for the query depicted in figure 2.

5 Similarity Assessment

The similarity assessor is handed a query and a set of Case Id's retrieved by the case selector, and sorts the corresponding cases by their degree of partial match to the query. Cases that match poorly are discarded. Similarity assessment divides naturally into two subtasks: assessing the pairwise similarity of individual low or high-pressure regions in a query and a case, and using cluster structure to combine those assessments into an overall assessment of goodness of match. We now briefly consider each of these subtasks.

Similarity assessment begins by computing the pairwise similarity of every region in the query and every region in the case. In assessing pairwise similarity, no distinction is made between regions internal to a cluster and the simple low or high-pressure regions at the cluster's leaves. To calculate the similarity of a pressure region in a query and a corresponding region in a case, MetVUW Workbench computes a weighted sum of a number of measures of similarity of shape, size, intensity, and degree of geographic displacement between the two regions.

MetVUW Workbench next uses these pairwise computations to calculate an overall goodness of match between clusters in a query and a case that also takes into account the degree of structural match between the clusters. The search enumerates sets of possible correspondences between regions in the query and the case, and returns the best set of correspondences that it finds. The system currently employs a form of width-bounded exhaustive search with successor ordering (Pearl, 1984).

During the search, the following structural constraints are imposed on the sets of correspondences. First, the relation that the correspondences define must be one-to-one. That is, no region in a query can correspond to more than one region in the case, and vice versa. Second, if a region r_q in the query corresponds to a region r_c in a case, then each child of r_q can only correspond to a child of r_c, and vice versa.

Similarity assessment is discussed in greater detail in (Jones and Roydhouse, 1994c).

6 Speeding Up Case Selection

As described in section 4, case selection proceeds in two stages. In stage 1, a series of *select/project* operations is issued, one for each weather system in a query. In stage 2, a *join* operation identifies those cases that match all weather systems in the query. Case selection can be expensive, especially for queries involving a large number of weather systems, because although each weather system in the query may match many cases in the case base, there may be few cases that match all the weather systems in the query. For complex queries such as these, the *select/project* operations can be needlessly slow, because much computational effort is expended to construct temporary relations most of whose tuples will eventually be revealed to be irrelevant. Likewise, the *join* operation can be slow, because the cost of a join is proportional to the size of the individual relations to be joined. There is no easy way to redesign the representation for cases to obviate the need for a series of *select/projects* and a *join*, because a case can contain an arbitrary number of high and low-pressure systems.

It is possible, however, to augment the case representation to make the *select/project* operations more focused and retrieve fewer irrelevant tuples. The key idea is to introduce a new "summary" field into the index labels for each weather system in the case base to encode key features of nearby regions: their type (e.g. high or low pressure), their approximate location relative to the weather system in question, and their approximate spatial extent. Each *select/project* operation at stage 1 of case selection then specifies both a particular weather system to retrieve and descriptions of any nearby regions in the query. The summary field in the index labels thus acts to quickly filter out tuples that would later be discarded anyway by the *join* at stage 2.

In this fashion, it is hoped to increase the efficiency of case selection. The efficiency of stage 1 should increase somewhat, because smaller temporary relations are constructed. The efficiency of stage 2 should increase significantly, because the relations that need to be joined together are smaller.

We use bit vectors are used to encode summary descriptions of nearby regions, because bit vectors can be efficiently matched in Postgres queries during stage 1 of case selection. For each region ρ of high or low pressure, a bit vector is allocated for each type of nearby region (e.g. high or low pressure). Each bit corresponds to a cell in a grid. The grid covers a region of the globe centred on ρ and is oriented along lines of latitude and longitude. Figure 6 illustrates the bit-vector representation for a region r_c of a hypothetical case. The bit vector is centred on r_c and encodes four nearby regions of high pressure, $h_c^1, \ldots, h_c^4$. A bit is set to 1 if a significant proportion of the corresponding cell in the grid is covered by a region of high pressure, otherwise the bit is set to 0. At the moment, "significant proportion" means that either at least 10% of the cell is covered by the region, or at least 10% of the region is contained in the cell. (The latter is required to do justice to very small regions such as h_c^4.)

During stage 1 of case selection, the various Postgres *select/project* operations are augmented to take advantage of these bit-vector summaries. For example, consider again the query from the user depicted in figure 2, above. Recall that this query involves three weather systems l_q, h_q^a, and h_q^b. To compose the *select/project* operation for region l_q, for example, two new bit vectors $B(l_q, h_q^a)$ and $B(l_q, h_q^b)$ are constructed, one for each other region in the query. The vector $B(l_q, h_q^a)$ represents a grid centred on l_q, a bit of which is set to 1 if and only if the region h_q^a covers a significant portion of the corresponding cell in the grid (see figure 7). The bit vector $B(l_q, h_q^b)$ is similar.

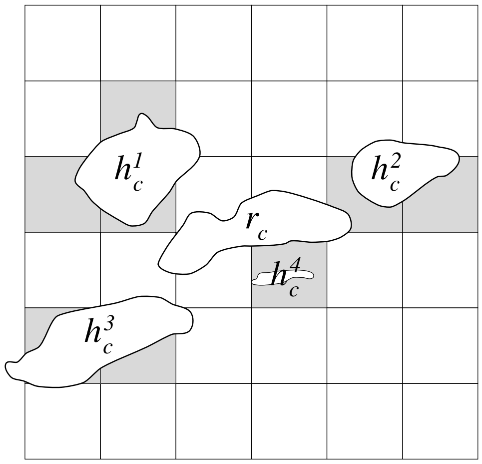


Figure 6: Graphical illustration of a bit-vector for the region r_c in a case. The shaded cells correspond to bits that are set to 1. In this example, the bit vector is `000000010000110011000100110000000000`. (Bits are allocated row by row, starting from the top left.)

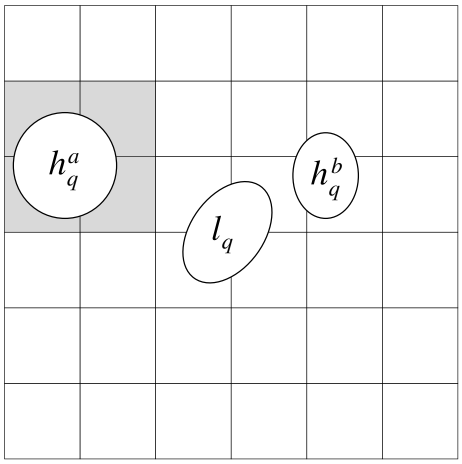


Figure 7: Graphical illustration of the bit vector $B(l_q, h_q^a)$ for the query of figure 2. In this example, the bit vector is `000000110000110000000000000000000000`.

The new Postgres *select/project* operation for the weather system l_q has the following form:

`select into R2 (R.case-id) from R in REGION_LOW where`
  bounding-box (l_q) `contains R.centre`
  `and R.bounding-box contains` *centre* (l_q)
  `and R.children >=` *children* (l_q)
  `and` $(B(l_q, h_q^a)$ `& R.bit-vector[HIGH])` $\neq$ `0`
  `and` $(B(l_q, h_q^b)$ `& R.bit-vector[HIGH])` $\neq$ `0`

Here, "`&`" represents bitwise conjunction, `r.bit-vector[`τ`]` picks out the bit vector centred on `R` for regions in the case of type τ, and `0` is the zero bit vector. The *select/project* operations for h_q^a and h_q^b are similar.

All the regions in a user's query are expanded by several degrees of latitude and longitude before bit-vector representations are computed, to allow a degree of displacement between the regions in the query and the corresponding regions in cases. Allowing for this displacement helps ensure that case selection doesn't filter out any cases that would be ranked as good matches by similarity assessment.

6.1 Empirical evaluation

A preliminary evaluation of the utility of bit-vector representations in case selection has been carried out, using three representative queries. The queries respectively request two, three, and four weather systems.

Table 2(a) gives average elapsed times for case selection for each query. The queries were run on a SparcStation 10 model 30 with 64 Mb of memory. Each query was run five times using the bit-vector representations and five times without using the bit-vector representations. The elapsed time for each query is broken down by stage of case selection. Table 2(b) shows the percentage improvement in time efficiency as a consequence of employing the bit-vector approach.

Query:	**Two**		**Three**		**Four**	
Vectors:	**no**	**yes**	**no**	**yes**	**no**	**yes**
Stage 1	7.9	8.1	11.4	11.9	30.4	27.4
Stage 2	3.4	3.2	6.9	4.2	12.4	3.6
Total	11.3	11.3	18.3	16.1	42.8	31.3

(a) Time in seconds

Query:	**Two**	**Three**	**Four**
Stage 1	-2%	-5%	10%
Stage 2	6%	39%	71%
Overall	0%	12%	27%

(b) Percentage improvement

Table 2: Case selection times and percentage improvement with and without bit-vector representations

The results show that as the number of weather systems specified by queries increases, the relative benefits of the bit vector summaries also increases. This is to be expected: when more weather systems are present in the query, more systems are available to restrict *select/project* operations. The breakdown by stage of case selection shows that these gains come mostly from improvements in the *join* operation at stage 2. The *select/project* operations begin to show some improvement for more complex queries, but actually degrade slightly for simpler queries.

These experimental results show that summary representations can greatly reduce the cost of joins in conjunctive queries for structured spatial information. In MetVUW Workbench, however, the impact on *select/project* operations is more limited, presumably because they spend most of their time locating tuples in memory and testing whether they should be included in the resulting temporary relation. They spend comparatively little time building the relation itself—that is, allocating space for the relation and copying tuples into it—because queries in MetVUW Workbench construct temporary relations containing only a single field, "Case Id." In a different application, queries might construct temporary relations containing multiple fields. For such applications, it is reasonable to expect that more of the database's time would be spent building the temporary relations, so the speedup afforded by summaries of spatial relations would be more significant.

7 Conclusion

Case-based reasoning is an increasingly important technology for developing knowledge-based systems (Kolodner, 1993). MetVUW Workbench demonstrates that a relational database can be used to manage a large case base. This approach to the design of case-based reasoning systems should find ready application in the environmental sciences, where knowledge bases are often very large. Two particular contributions of our approach are as follows:

An architecture for case retrieval.
Case retrieval in MetVUW Workbench is split into separate processes of case selection and similarity assessment. Case selection uses the efficient indexing and retrieval machinery of the relational database to quickly identify candidate cases for retrieval. Similarity assessment then analyses these cases in more detail, using a process of knowledge-intensive partial matching.

This general architecture for case-based reasoning systems should prove useful in a wide range of large scale case-based reasoning applications.

2. Summaries of spatial relationships.
A field summarising the spatial relationships between each weather system and other nearby weather systems is stored as part of each index label. This summary field serves to speed up case selection. Summaries of this kind can be used to increase the efficiency of any retrieval task involving queries for sets of objects with spatial extent that bear some spatial relationship to one another.

Acknowledgements

This research was carried out in collaboration with James McGregor of the Institute of Geophysics at Victoria University of Wellington. The idea of using spatial relations to focus case selection arose during discussions with Peter Andreae and David Andreae. Our thanks to Erick Brenstrum of the Meteorological Service of New Zealand for useful discussions regarding the meteorologically significant features of low and high pressure systems. The Meteorological Service of New Zealand provided the situation maps in figure 3.

References

Domeshek, E. A. (1991). Indexing stories as social advice. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 16–21. AAAI Press.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM-SIGMOD Conference on Management of Data*. ACM-SIGMOD.

Hunter, L. E. (1989). *Knowledge Acquisition Through Experience*. PhD thesis, Yale University.

Jones, E. K. and Roydhouse, A. (1994a). Intelligent retrieval of historical meteorological data. *AI Applications*, 8(3):43–54.

Jones, E. K. and Roydhouse, A. (1994b). Iterative design of case retrieval systems. In *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning*, pages 150–156. Also available as Technical report CS-TR-94/6, Victoria University of Wellington. URL: ftp://ftphost.comp.vuw.ac.nz/doc/vuw-publications/CS-TR-94/CS-TR-94-6.ps.gz.

Jones, E. K. and Roydhouse, A. (1994c). Spatial representations of meteorological data for intelligent retrieval. In *Proceedings of the Sixth Colloquium of the Spatial Information Research Centre, University of Otago, New Zealand*, pages 45–58.

Kolodner, J. L. (1991). Improving human decision making through case-based decision aiding. *AI Magazine*, 12(2):52–68.

Kolodner, J. L. (1993). *Case-based reasoning*. Morgan Kaufmann, San Mateo, California.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, Massachusetts.

Riesbeck, C. K. and Schank, R. S. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum Assoc. Inc., Hillsdale, New Jersey.

Stonebraker, M. and Rowe, L. (1986). The design of Postgres. In *Proceedings of the 1986 SIGMOD Conference on Management of Data*. ACM Press.