Parallelization of a Finite Element Surface Fitting Algorithm for Data Mining

Peter Christen* Irfan Altas†
Markus Hegland and Stephen Roberts‡
Kevin Burrage and Roger Sidje§

May 31, 2000

Abstract

A major task in data mining is to develop automatic techniques to process and to detect patterns in very large data sets. An important data mining technique is multivariate regression, and an essential sub task is the estimation of interaction surfaces, i.e. the estimation of functions of two variables. Thin plate splines provide a very good method to determine an approximating surface. Obtaining standard thin plate splines requires the solution of a dense linear system of equations of order n, where n is the number of observations. Standard thin plate splines may not be practical, because the number of observations for data mining applications is often in the millions. We have developed a finite element approximation of a spline that can handle data sizes with millions of records. The resolution of the finite element method can be chosen independently from the number of observations. The observation data is read from secondary storage once, and does not need to be stored in memory. In this paper, we present a first parallel implementation of this method in an MPI environment.

*Computer Science Laboratory, RSISE, Australian National University, Canberra, ACT 0200, Australia, Peter.Christen@anu.edu.au

†School of Information Studies, Charles Sturt University, Wagga Wagga, NSW 2678, Australia, ialtas@csu.edu.au

‡Computer Science Laboratory, RSISE, Australian National University, Canberra, ACT 0200, Australia, Markus.Hegland@anu.edu.au, Stephen.Roberts@anu.edu.au

§Department of Mathematics, University of Queensland, St. Lucia, QLD 4072, Australia, kb@maths.uq.edu.au, rbs@maths.uq.edu.au

1 Introduction

In the last decade, there has been an explosive growth in the amount of data being collected. The computerisation of business transactions and use of bar codes in commercial outlets have provided businesses with enormous amounts of data. Revealing patterns and relationships in a data set can improve the goals, missions and objectives of many organisations. For example, sales records can reveal highly profitable retail sales patterns. As such, it is important to develop automatic techniques to process and to detect patterns in very large data sets [1]. This process is known as *Data Mining*.

Data mining techniques are used to spot trends in data that may not be easily detectable by traditional database query tools that rely on simple queries to produce results. Data mining tools reveal hidden relationships and patterns as well as uncover correlations that are not detected by simple database queries [3].

An important technique applied in data mining is multivariate regression which is used to determine functional relationships in high dimensional data sets. A major difficulty which one faces when applying nonparametric methods is that the complexity grows exponentially with the dimension of the data set. This has been called the *curse of dimensionality*. Additive and interaction splines can be used to overcome this curse [4]. In the case where interaction terms in the splines are limited to order two interactions, the model consists of a sum of functions of one or two variables and the fitting problem thus is reduced to fitting a sum of functions of two variables. One could call this problem surface fitting of a set of coupled surfaces. As such, surface fitting is an important technique for the data mining of large data sets.

We have developed a generic surface fitting algorithm that can handle data sizes with millions of observations. The algorithm combines the favourable properties of finite element surface fitting with the ones of thin plate splines. An overview of this *thin plate spline finite element method* (TPSFEM) is presented in Section 2. We discuss the numerical solution of the linear equations arising from the TPSFEM in Section 3. In order to interactively analyze large data sets, a considerable amount of computational power is needed in most cases. High-performance parallel and distributed computing are crucial for ensuring system scalability and interactivity as data sets grow considerably in size and complexity. The parallel implementation of the algorithm is presented in Section 4 whereas first test results are given in Section 5. Conclusions and future work are discussed in Section 6.

2 Surface Fitting Algorithm

Surface fitting and smoothing splines techniques are widely used to fit data. We have developed a surface fitting algorithm, TPSFEM, that can be used for various applications such as data mining and digital elevation models.

The TPSFEM algorithm can handle data with millions of records. It combines the favourable properties of finite element surface fitting with the ones of thin plate splines. The standard thin plate spline is the function f_α that minimises the functional

$$M_\alpha(f) = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}^{(i)}) - y_i)^2 + \alpha \int_\Omega \left(\left(\frac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} \right)^2 + 2 \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(\mathbf{x})}{\partial^2 x_2} \right)^2 \right) d\mathbf{x}$$

where the observations of the predictor and response variables, respectively are given by $\mathbf{x}^{(i)} \in \mathbb{R}^2$, and $y^{(i)} \in \mathbb{R}$ for $i = 1, ..., n$. An appropriate value for the smoothing parameter α can be determined by generalised cross validation. The smoothing problem will be solved with finite elements.

In order to reduce the complexity of the problem we suggest that very simple elements are used, namely, tensor products of piecewise linear functions. For these functions, however, the required second derivatives do not exist. Thus one has to use a non-conforming finite element principle and a new functional is introduced which only consists of second derivatives. The interested reader can refer to [2, 5, 6] for a detailed derivation of the method.

After some manipulations and discretization [6] one gets the following linear system of equations which describes the finite element solution of the minimisation problem:

$$\begin{bmatrix} \alpha A & 0 & 0 & 0 & -B_1^T \\ 0 & \alpha A & 0 & 0 & -B_2^T \\ 0 & 0 & M & F & A \\ 0 & 0 & F^T & E & 0 \\ -B_1 & -B_2 & A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_0 \\ \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ n^{-1}N^T\mathbf{y} \\ n^{-1}X^T\mathbf{y} \\ 0 \end{bmatrix} \tag{1}$$

The symmetric positive definite matrix A is the finite element approximation. $\mathbf{c} = [c_0\ c_1\ c_2]^T \in \mathbb{R}^3$ is constant and $\mathbf{w}$ is the Lagrange multiplier vector. $\mathbf{u}_0$ is a discrete approximation to f_α whereas $\mathbf{u}_1$ and $\mathbf{u}_2$ are the discrete approximation of the first derivatives of f_α.

3 Solution of Linear Systems

The size of the linear system (1) is independent of the number of observations, n. The observation data are visited only once during the assembly of the matrices M, E and F and the vectors $N\mathbf{y}$ and $X\mathbf{y}$ in (1). Thus, the time complexity to form (1) is $O(n)$. If the number of nodes in the finite element discretization is m, the size of (1) is $4m+3$ which is independent of n. All subsystems in (1) have dimension m, except the fourth one which has dimension 3. Thus, the total amount of work required for the TPSFEM algorithm is $O(n)$ to form (1) plus the work required to solve it. The sequential time can be modelled as

$$T_{total} = n\Delta T_{assembly} + T_{solve}(m)$$

with $\Delta T_{assembly}$ the time to process one single data point and $T_{solve}(m)$ the solver time, which depends upon the size of the linear system.

It can be seen that the system matrix (1) is symmetric indefinite and sparse. One of the efficient techniques to solve such systems is the Uzawa algorithm [7, 8]. We have applied a modified version of the Uzawa algorithm to (1). If several adjacent elements do not contain any observation points, the matrix M in (1) becomes singular, hence, the Uzawa algorithm becomes inapplicable. In order to avoid this problem we make a block row interchange of the third and fifth block row of the system (1). Then, the variant of the Uzawa algorithm used here has the form of a block Gauss-Seidel iteration and can be written as

$$\begin{bmatrix} \alpha A & 0 & 0 & 0 & 0 \\ 0 & \alpha A & 0 & 0 & 0 \\ -B_1 & -B_2 & A & 0 & 0 \\ 0 & 0 & F^T & E & 0 \\ 0 & 0 & M & F & A \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^{k+1} \\ \mathbf{u}_2^{k+1} \\ \mathbf{u}_0^{k+1} \\ \mathbf{c}^{k+1} \\ \mathbf{w}^{k+1} \end{bmatrix} = \begin{bmatrix} B_1^T \\ B_2^B \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathbf{w}^k + \begin{bmatrix} 0 \\ 0 \\ 0 \\ n^{-1}X^T\mathbf{y} \\ n^{-1}N^T\mathbf{y} \end{bmatrix} \qquad (2)$$

This system requires the solution of four equations with matrix A for each iteration step. The convergence behaviour and some variations of this approach are discussed in [6]. In the next section, we discuss the parallel implementation.

4 Implementation and Parallelization

The original implementation of the TPSFEM algorithm has been done in Matlab [5, 6]. We have chosen C and MPI [9] for a second implementation. The main purpose of this implementation is to analyze parallelization aspects of the TPSFEM algorithm and increase performance. The matrices needed in (2) are sparse and consist of nine diagonals with non-zero elements. As matrices M and A are symmetric, only the diagonal and upper part have to be stored. Matrices B_1 and B_2 are non-symmetric, so all nine diagonals are stored. We have chosen a diagonal storage data structure, consisting of a two-dimensional array with m rows and 5 or 9 columns, respectively. Each of the columns contains one of the matrix diagonals with non-zero elements. With this packed data structure, good data locality and thus good cache utilization can be achieved on RISC machines. The matrix F is of dimension $m \times 3$ and matrix E is a small 3×3 matrix. The dot product and vector addition (DAXPY) implementations use loop unrolling for better RISC pipeline usage.

As the data sets in data mining applications can be quite large, file reading and assembly of the matrices M, F and E as well as the vectors $N\mathbf{y}$ and $X\mathbf{y}$ are time consuming steps. Fortunately, in the TPSFEM algorithm the data has to be read only once and the process of assembling the matrices can be parallelized easily. First, the data file is split equally into P smaller files by a cyclic distribution. These files then can be processed independently by P processors. Each of them assembles local matrices and vectors. After the assembly, these local matrices and vectors have to be collected and summed to get the final matrices M, F, E and vectors $N\mathbf{y}$ and $X\mathbf{y}$. The amount of data to be communicated depends on m, but not on the number of observations n. An almost ideal speedup can be achieved for the assembly process, if the amount of communicated data is small compared to the number of observations to process from file. The assembly of the matrices A, B_1 and B_2 only depends on the matrix dimension m and takes much less time than the assembly of M, F and E.

As the coefficient matrix A of the four large sub-systems is symmetric positive definite, they can be solved with the Conjugate Gradient method. The fourth sub-system is so small we can solve it directly. The first and second sub-systems are independent, so they can be solved in parallel. Our first parallel implementation applies a functional parallelism in the following way:

- The assembly of matrices M, F, E and vectors $N\mathbf{y}$ and $X\mathbf{y}$ is done by all P processors, where each of them reads one locally stored data file with n/P observations. After the assembly, the local matrices and vectors are collected and summed on the host processor P_0 (with a simple call to `MPI_Reduce`). Two messages are communicated, one containing M ($5m$ floating point values), the other one containing F, E, $N\mathbf{y}$ and $X\mathbf{y}$ (with $4m + 12$ floating point values).
- The matrices A, B_1 and B_2 are assembled on processors P_0 and P_1.
- The iterative solving with the Uzawa's algorithm is only started on P_0 and P_1. After the initialization, processor P_0 solves the first sub-

system in (2). Simultaneously, processor P_1 starts solving the second sub-system in (2) and sends the resulting vector $\mathbf{u}_2$ to processor P_0, which then can solve the remaining sub-systems. The solution cost of the small fourth sub-system is negligible. After the convergence check, processor P_0 sends the vector $\mathbf{w}$ to P_1 which then starts the next solving of the second sub-system.

Applying Amdahl's law, the solver part of the algorithm can achieve at most a speedup of about 1.333 if the communication of vectors is neglected and all four large sub-systems need the same time to be solved. So the total time for this parallel version of the TPSFEM algorithm on P processors becomes:

$$T_{total} = \frac{n}{P}\Delta T_{assembly} + \frac{3}{4}T_{solve}(m)$$

This shows that the algorithm scales linearly with n. For the assembly step, using twice as many processors allows us to process data sets with twice as many observations in the same time.

5 Tests Results from Parallel Implementation

The TPSFEM algorithm has already been applied to fit surfaces to large data sets from insurance, flow field, digital elevation and magnetic field areas. In this section, we demonstrate the efficiency of the parallel implementation of the TPSFEM algorithm by using two large digital elevation data sets. The algorithm can equally be applicable as a nonparametric regression technique used in data mining.

Elevation data is an important data component required to transform a Geographical Information System (GIS) from a 2-dimensional map storage system into a 3-dimensional information system [10]. Hence, an efficient surface fitting algorithm such as TPSFEM is an important tool for GIS applications. The involvement of end-users in this area is usually interactive. Therefore, they can benefit significantly from a fast surface fitting algorithm.

Two digital elevation data sets are obtained by digitizing the map of the Murrumbidgee region in Australia. They have $n = 547453$ and $n = 1887250$ observation points, respectively. The test platform are 10 Sun Sparc-5 workstations connected by a 10 Mbit/s twisted pair Ethernet. All timings presented here are average of ten test runs on an otherwise idle platform.

The first table shows timings in milliseconds for the assembly stage of the first data set. We did the test runs for three different matrix dimensions. One can clearly see how the computation part is independent of the matrix dimension, but the communication costs increase dramatically with increased matrix dimensions.

m	Serial Assembly	Parallel Assembly on 10 Processors		
		Computation	Communication	Total
2'601	87'159	8'781	1'258	10'039
10'201	86'809	8'770	5'162	13'932
63'001	87'250	8'935	29'827	38'762

In the second table we present timings for the second data set with $m = 2'601$. The serial assembly and solution steps took $285'780\ ms$ and $40'893\ ms$, respectively. The parallel solution step with two processors is $32'384\ ms$.

Number of Processors	Parallel Assembly		
	Computation	Communication	Total
3	97'231	1'123	98'354
7	41'653	1'524	43'177
10	28'187	1'800	30'987

Finally, in the third table we introduce timings for the second data set with $m = 10'201$. The serial assembly and solution steps took $284'939\ ms$ and $624'287\ ms$, respectively. The parallel solution step with two processors is $548'180\ ms$.

Number of Processors	Parallel Assembly		
	Computation	Communication	Total
3	98'143	1'762	99'905
7	41'856	4'382	46'238
10	29'374	5'769	35'143

6 Conclusions

In this work we presented a first parallel version of the TPSFEM algorithm, which can handle very large data sets to fit smooth surfaces. There are two main parts of the algorithm that consume most processing time: Assembly of the matrices in (1) and solving the system (2). For very large data sets an almost ideal speedup can be achieved in the assembly step. Unfortunately, the functional parallelism in the solver part of this implementation is not scalable, in contrast to the assembly part. Our future research therefore will concentrate on how to achieve scalability by using distributed algorithms to solve the linear systems. In order to solve the system (1) we are developing a generalized cross validation technique.

7 Acknowledgment

This research was supported by the Advanced Computational Systems CRC. Peter Christen's stay in Australia has been made possible by grants from the *Swiss National Science Foundation* (SNF) and the *Novartis Stiftung, vormals Ciba-Geigy Jubiläums-Stiftung.*

References

[1] Data Mining: An Introduction, Data Distilleries, DD19961, 1996, `http://www.ddi.nl`

[2] P. Christen, I. Altas, M. Hegland, S. Roberts, K. Burrage and R. Sidje, A Parallel Finite Element Surface Fitting Algorithm for Data Mining, submitted to Parco99, Delft, Holland.

[3] A.A. Freitas and S.H. Lavington, Mining Very Large Databases with Parallel Processing, Kluwer Academic Publishers, 1998.

[4] T.J. Hastie and R.J. Tibshirani, Generalized Additive Models, Monographs Chapman and Hall, 1990.

[5] M. Hegland, S. Roberts and I. Altas, Finite Element Thin Plate Splines for Data Mining Applications, in Mathematical Methods for Curves and Surfaces II, M. Daehlen, T. Lyche and L.L. Schumaker, Eds., pp. 245-253, Vanderbilt University Press, 1998, `ftp://farrer.riv.csu.edu.au:/pub/irfan/hegra97a.ps.gz`

[6] M. Hegland, S. Roberts and I. Altas, Finite Element Thin Plate Splines for Surface Fitting, in Computational Techniques and Applications: CTAC97, B.J. Noye, M.D. Teubner and A.W. Gill, Eds., pp. 289-296, World Scientific, 1997, `ftp://farrer.riv.csu.edu.au:/pub/irfan/hegra97b.ps.gz`

[7] M. Fortin and R. Glowinski, Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems, North-Holland, 1983.

[8] H.C. Elman and H.G. Golub, Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems, SIAM J. Numer. Anal., **31**, 1994, pp. 1645-1661.

[9] W. Gropp, E. Lusk and A. Skjellum, Using MPI – Portable Parallel Programming with the Message-Passing Interface, The MIT Press, Cambridge, Massachusetts, 1994.

[10] L. Lang, GIS Goes 3D, Computer Graphics World, March 1989, pp. 38-46.