J. Appl. Math. & Informatics Vol. **28***(2010), No. 1 - 2, pp. 351 - 361*
Website: http://www.kcam.biz

DATA MINING AND PREDICTION OF SAI TYPE MATRIX PRECONDITIONER

SANG-BAE KIM,* SHUTING XU AND JUN ZHANG

ABSTRACT. The solution of large sparse linear systems is one of the most important problems in large scale scientific computing. Among the many methods developed, the preconditioned Krylov subspace methods are considered the preferred methods. Selecting a suitable preconditioner with appropriate parameters for a specific sparse linear system presents a challenging task for many application scientists and engineers who have little knowledge of preconditioned iterative methods. The prediction of ILU type preconditioners was considered in [27] where support vector machine(SVM), as a data mining technique, is used to classify large sparse linear systems and predict best preconditioners. In this paper, we apply the data mining approach to the sparse approximate inverse(SAI) type preconditioners to find some parameters with which the preconditioned Krylov subspace method on the linear systems shows best performance.

AMS Mathematics Subject Classification :65F10, 65F50, 68P20
Key words and phrases : Sparse matrix, preconditioning, sparse approximte inverse, support vector machine

1. Introduction

The solution of large sparse linear systems is one of the most important problems in large scale scientific computing. For the past 50 years, many direct and iterative methods have been developed for this purpose[7, 16]. Among them, preconditioned Krylov subspace methods[16] with a Krylov iterative solver and a preconditioner are considered the preferred methods. The preconditioners employed in the preconditioned iterative solvers usually determine the overall convergence rate of the iterative procedure[28]. However, selecting a suitable preconditioner for a specific sparse matrix arising from a particular application to achieve fast convergence is the combination of art and science, and presents

Received August 14, 2009. Accepted November 17, 2009. *Corresponding author.
The researh work of S. B. Kim was supported in part by Hannam University Research Fund 2006.

a challenging problem for many application scientists and engineers who have little knowledge of the preconditioned iterative methods[11, 12, 14].

There is an enlarging gap between the development of more and more sophisticated preconditioned iterative solvers by the computational linear algebra community and the ability to understand and to properly use these solvers by the application scientists and engineers to solve their more and more complex modeling and simulation problems. High performance computers and numerical algorithms will be less useful if they are not matched with the intended application problems. In the context of preconditioning, the use of a wrong preconditioner may cause an iteration process to diverge.

There has been a considerable amount of effort made by several researchers and organizations to collect various sparse matrices in order to use them or for test purposes. The National Institute of Standard and Technology (NIST) has been playing a leading role in this endeavor and currently hosts one of the largest such repositories: MatrixMarket[15]. Several other collections have been contributed by engineers, scientists and numerical analyst, e.g., the well-known Harwell-Boeing sparse matrix collection and the University of Florida sparse matrix collections[6]. NIST has done some categorization work and published some preliminary information on these matrices. For each matrix this information includes its type, dimensions, condition number, nonzero structure, etc. MatrixMarket is becoming a standard source of sparse matrices for testing various direct and iterative solution methods. However, there is no information regarding which matrix can be solved by what method using what parameters. Such information would be extremely helpful for application scientists and engineers as it would enable them to choose suitable sparse matrix solvers for certain class of applications.

It is attractive to use machine learning techniques to help application scientists and engineers to choose suitable preconditioners for their particular application problems. Sparse matrices arising from different applications do have certain different features. These features may be represented by the sizes and the locations of their nonzero entries. If we can determine and extract these matrices features, and study and learn how the performance of the preconditioned Krylov subspace methods is related to these matrix features, we may be able to predict the performance of these preconditioned iterative methods to solve other sparse matrices that may have the same or similar features.

The idea of using matrix features and data mining techniques to predict the possibility of solving a sparse matrix by some preconditioners was first proposed in [29]. The process of extracting sparse matrix features and the identified 66 matrix features are described in [20, 22, 21]. Data mining techniques with matrix features were applied for predicting the condition numbers of sparse matrices [23, 25] and for predicting the solving status of sparse matrices by matrix structure-based incomplete LU type preconditioners such as ILU(0), ILU(k) and ILUT [24, 26, 27]. ILUT is different from ILU(0), ILU(k) in point of view that it needs two preset parameters and it only works well under some sets of values of these parameters. Different sparse linear systems usually can be solved with different ILUT parameters.

In this paper, we study another type of preconditioner, i.e. sparse approximate inverse (SAI) [4, 8] to solve sparse linear systems. SAI preconditioner has two preset parameters like ILUT. These two parameters are related to the sparsity pattern and filtering tolerance which will be explained in Section 2. Our aim is to predict with which parameter sets the sparse systems can by solved by SAI. The SAI preconditioners possess high degree of parallelism while the ILU [16] is inherently sequential in both the construction and the application phases. The performance of SAI preconditioner depends on the choice of the sparsity pattern. The static and dynamic sparsity pattern algorithms have been proposed [4, 5, 8, 13]. We apply only the static sparsity pattern scheme because the dynamic sparsity pattern algorithm changes the parameter in the process of solver iteration, while we want to find some a priori parameters of SAI preconditioner with which the iterative solver achieves best performance.

This paper is organized as follows. In Section 2, we explain the concept of sparse approximate inverse preconditioner. We can see what kind of parameters are involved in the preconditioner. Section 3 describes the parameter space relating to the SAI preconditioner. The experiments are carried out and the results are reported in Section 4. The conclusion of this paper is in Section 5.

2. Sparse approximate inverse preconditioner

In order for a preconditioner to be efficient, the construction of the preconditioning matrix M should be cheap and the inverse of M or the solution with the matrix M should be inexpensive. On parallel computers, it is ideal that both the preconditioner construction (setup) phase and the preconditioner application (solution) phase posses high degree of parallelism. So the main difficulty in front of us is to find a high quality preconditioner with good parallelism so that the large linear problems can be solved efficiently on the high performance distributed memory parallel computers.

There are many methods to compute the preconditioner M. One popular method is called the incomplete LU factorizations (ILU(k), ILUT) [16]. The ILU-type preconditioning techniques try to compute an approximation of the original matrix A and transform the linear system

$$Ax = b, \tag{1}$$

into

$$(LU)^{-1}Ax = (LU)^{-1}b. \tag{2}$$

Here the preconditioner LU consists of a lower triangular matrix (L) and an upper triangular matrix (U), and $LU \approx A$. The ILU-type preconditioner is widely used and shown to be efficient and effective for certain kind of problems. However, since the ILU preconditioners are based on various Gauss elimination

[16], they are inherently sequential in both the construction and the application phases and are difficult to implement on parallel platforms.

The sparse approximate inverse (SAI) preconditioner is an interesting alternative of ILU-type preconditioner [1, 2]. Instead of computing an approximation of A, the SAI-type preconditioning techniques try to compute a sparse approximation of A^{-1} directly. In this case, the preconditioned system is

$$MAx = Mb, \tag{3}$$

where $M \approx A^{-1}$.

The sparse approximate inverse technique is based on the idea of computing a preconditioning matrix M which approximates A^{-1} in the Frobenius norm [4, 8]. Since we want M to be a good approximation to A^{-1}, it is ideal if $MA \approx I$. This approach is to approximate A^{-1} from the left, and M is called the left preconditioner. It is also possible to approximate A^{-1} from the right, so that $AM \approx I$, which is termed as the right preconditioner. In the case of the right preconditioning, the equivalent preconditioned system is

$$AMy = b, \qquad and \qquad x = My. \tag{4}$$

In fact, the right preconditioning approach is easier for us to illustrate the Frobenius norm minimization idea, which will be described in detail in the following paragraphs.

In order to have $AM \approx I$, we want to minimize the functional

$$f(M) = \min_{M} \|AM - I\| \tag{5}$$

for all possible nonsingular square matrices M of order n, with respect to a certain norm. Without any constraint on M, the minimization problem (5) has an obvious solution, i.e., $M = A^{-1}$. This obvious solution is undesirable for at least two reasons. First, the computational cost for solving the unconstrained minimization problem (5) is prohibitively high. Second, for most sparse matrices A, their inverses A^{-1} are dense, which will cause memory problems for large scale matrices encountered in many practical applications.

Thus we are interested in a constrained minimization such that M has a certain sparsity pattern (nonzero structure), i.e., only certain entries of M are allowed to be nonzero. Given a set of sparsity patterns Ω, we minimize the functional

$$f(M) = \min_{M \in \Omega} \|AM - I\|. \tag{6}$$

Although any norm can potentially be used in the above definition, a particularly convenient norm is the Frobenius norm which is defined for a matrix $A = (a_{ij})_{n\times n}$ as $\|A\|_F = \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}$ [16]. With the Frobenius norm, the minimization problem (6) is decoupled into n independent subproblems and can proceed as (using square for convenience)

$$\|AM - I\|_F^2 = \sum_{k=1}^{n} \|(AM - I)e_k\|_2^2 = \sum_{k=1}^{n} \|Am_k - e_k\|_2^2, \tag{7}$$

where m_k and e_k are the kth column of M and that of I, respectively. It follows that the minimization problem (6) is equivalent to minimizing the individual functions

$$\|Am_k - e_k\|_2, \qquad k = 1, 2, \ldots, n \tag{8}$$

with certain restrictions placed on the sparsity pattern of m_k. In other words, each column of M can be computed independently. For the moment, we assume that the sparsity pattern of m_k is given, i.e., a few, say n_2, entries of m_k at certain locations are allowed to be nonzero, the rest of the entries are forced to be zero. Denote the n_2 nonzero entries of m_k by $\tilde{m}_k$ and the n_2 columns of A corresponding to $\tilde{m}_k$ by A_k. Since A is sparse, the submatrix A_k has many rows that are identically zero. After removing the zero rows, we have a reduced matrix $\tilde{A}_k$ with n_1 rows. The individual minimization problem (8) is reduced to a least squares problem of order $n_1 \times n_2$

$$\min_{\tilde{m}_k} \|\tilde{A}_k \tilde{m}_k - \tilde{e}_k\|_2, \qquad k = 1, 2, \ldots, n. \tag{9}$$

We note that the matrix $\tilde{A}_k$ is usually a very small rectangular matrix. It has full rank if A is nonsingular.

There are a variety of methods available to solve the least squares problem (9). One approach, proposed by Grote and Huckle [8], is to solve (9) using a QR factorization as

$$\tilde{A}_k = Q_k \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \tag{10}$$

where R_k is a nonsingular upper triangular $n_2 \times n_2$ matrix. Q_k is an $n_1 \times n_1$ orthogonal matrix, such that $Q_k^{-1} = Q_k^T$. The least squares problem (9) is solved by first computing $\tilde{c}_k = Q_k^T \tilde{e}_k$ and then obtaining the solution as $\tilde{m}_k = R_k^{-1} \tilde{c}_k(1 : n_2)$. In this way, $\tilde{m}_k$ can be computed for each $k = 1, 2, \ldots, n$, independently. This yields an approximate inverse matrix M, which minimizes $\|AM - I\|_F$ for the given sparsity pattern.

The inherent parallelism is obvious in the process of computing $\tilde{m}_k$ independently of each other. Its performance depends on the choose of the sparsity pattern. The static and dynamic sparsity pattern algorithms have been proposed [4, 5, 8, 13]. Generally people think the dynamic sparsity pattern method is more robust but with more computational costs than the static sparsity pattern method. The comparative study between two algorithms can be found in [19]. The dynamic sparsity pattern method can not be applied for our study of the prediction of SAI preconditioner because the SAI preconditioner is modified in the process of solver iteration. In our study, the parameters should be assigned a priori.

The static sparsity pattern strategies use a priori patterns, e.g., the banded pattern [2, 9]. A particularly useful and effective strategy is to use the sparsity pattern of the matrix A or A^T. Chow [3] proposed to use sparsified patterns of A as the sparsity pattern for M. Here "sparsified" means that certain small entries of A are removed before its sparsity pattern is extracted. The following SAI algorithm with the static sparsity pattern is revised from the one in [3].

ALGORITHM **2.1.** *Construct a static pattern sparse approximate inverse preconditioner.*

```
1.  Given a drop tolerance $\tau$ and $\epsilon$
2.  Sparsify $A$ with respect to $\tau$
3.  Compute a sparse approximate inverse $M$ according to
    the sparsity pattern of $A$
4.  Drop small entries of $M$ with respect to $\epsilon$
5.  $M$ is the preconditioner for $Ax = b$
```

3. SAI Parameter Space

We use the preconditioned GMRES (PGMRES) as our choice of preconditioned Krylov subspace method, i.e., we use the iterative solver GMRES with the SAI preconditioner. PGMRES with SAI preconditioner can solve some sparse linear systems that would fail with other preconditioners like ILUT. According to Algorithm 2.1, there are two parameters are involved in computing our SAI preconditioner. The first one is the dropping tolerance τ by which the relatively small entries of matrix A in Equation (6) with respect to the corresponding diagonal entries are dropped to get a new matrix A'. The second one ϵ is for dropping the relatively small entries of M with respect to the corresponding diagonal entries after computing $M \approx A'^{-1}$. There exist numerous SAI algorithms. We may adopt different parameter sets which depend on the SAI algorithms.

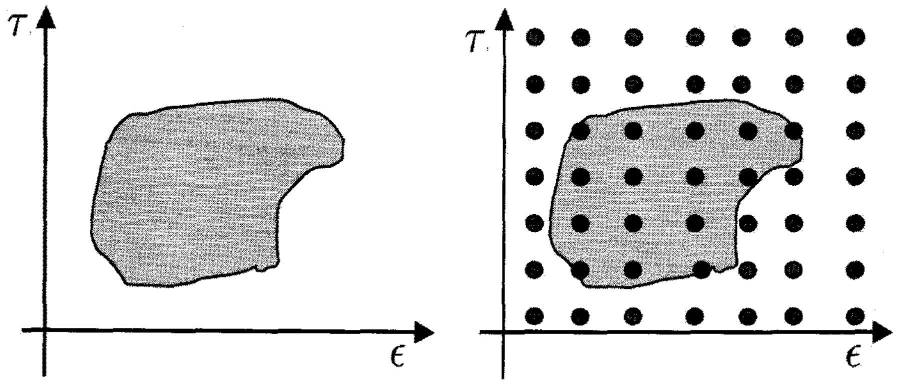

FIGURE 1. Parameter Space of SAI.

Whether a sparse linear system can be solved by SAI and the number of iterations the PGMRES will take are closely related to the values of these two parameters. Given a sparse linear system, we want to predict all the possible combination of the parameters with which the linear system can be solved. The possible combination of the parameters may construct arbitrary areas in a two-dimensional parameter space. As the case of ILUT [27], we choose some points in the parameter space as samples (refer to Figure 1). Studying the performance of SAI with these sample parameters, we can get the main idea of what kinds of combination of the parameters are favorable for a given sparse system. If we can correctly predict the solving status of a sparse linear system at the sample points, we may obtain an outline of the parameter area(s) in which the sparse linear system can be solved.

We use support vector machine (SVM) classification to predict the solving status of the sparse linear systems by SAI with a specific set of parameters. The explanation how the SVM can be applied for the prediction is found in [24, 27].

4. Experiment and Result

We conduct some experiments to test the prediction accuracy of the solving status of the 316 sparse linear systems by PGMRES with preconditioner SAI with different parameter sets. The sparse linear systems are constructed by using sparse matrices from MatrixMarket [15]. The parameter setting in PGMRES is the same as that used in [27]. To be specific, the right hand sides of the linear systems are constructed by assuming that the solutions are a vector of all ones. The initial guessed solutions are a vector of all zeros. The maximum number of iterations is 500. The convergence stopping criterion is that the 2-norm of the residual vectors is reduced by 7 orders of magnitude. The iterative method used is GMRES(20) and the preconditioner is SAI. We test the accuracy of predicting the combination of parameters using SVM classification [18, 17]. We use SVM^{Light} [10] for SVM classification software. The kernel we used in SVM classification is RBF[10]. The results are obtained by using a 5-fold cross validation. Detailed description about the SVM classification and its applications to the matrix preconditioner prediction can be found in [20, 24, 27]. For the parameters τ and ϵ, we choose the most often used values 0.1, 0.01, 0.001, 0.0001, 0.00001.

Table 1 shows the average prediction accuracy with different combination of the parameters when the σ in RBF kernel is set to 0.1. We see that the highest prediction accuracy 92.4% is obtained at four points around $(\tau, \epsilon) = (0.00001, 0.00001)$. The lowest prediction accuracy 88.0% is obtained with $(\tau, \epsilon) = (0.01, 0.01)$. Generally speaking, the prediction accuracy reaches the highest value around the point $(\tau, \epsilon) = (0.00001, 0.00001)$.

Table 2 also shows the average prediction accuracy with different combination of parameters. The difference from Table 1 is that σ is set to be 0.01 in RBF kernel. It shows a different distribution of prediction accuracy to Table 1 and

$\tau\backslash\epsilon$	0.1	0.01	0.001	0.0001	0.00001
0.10000	0.892405	0.895570	0.911392	0.905063	0.895570
0.01000	0.889241	0.879747	0.895570	0.892405	0.901899
0.00100	0.886076	0.901899	0.898734	0.914557	0.914557
0.00010	0.892405	0.908228	0.895570	**0.924051**	**0.924051**
0.00001	0.892405	0.917722	0.892405	**0.924051**	**0.924051**

TABLE 1. Prediction accuracy of SVM classification ($\sigma = 0.1$).

$\tau\backslash\epsilon$	0.1	0.01	0.001	0.0001	0.00001
0.10000	0.892405	0.892405	0.898734	0.901899	0.905063
0.01000	0.905063	0.901899	0.914557	**0.920886**	**0.920886**
0.00100	0.901899	0.911392	0.917722	0.911392	0.911392
0.00010	0.889241	0.917722	0.908228	0.917722	0.917722
0.00001	0.889241	0.917722	0.895570	0.917722	0.917722

TABLE 2. Prediction accuracy of SVM classification ($\sigma = 0.01$).

$\tau\backslash\epsilon$	0.1	0.01	0.001	0.0001	0.00001
0.10000	0.908228	0.911392	0.914557	**0.930380**	0.905063
0.01000	0.917722	0.908228	0.914557	0.920886	0.927215
0.00100	0.905063	0.914557	0.920886	0.927215	0.927215
0.00010	0.898734	0.911392	0.917722	**0.930380**	**0.930380**
0.00001	0.898734	0.908228	0.889241	**0.930380**	**0.930380**

TABLE 3. Prediction accuracy of SVM classification ($\sigma = 0.001$).

the highest prediction accuracy is achieved around point $(\tau, \epsilon) = (0.01, 0.00001)$ with prediction accuracy 92.1%.

Table 3 is obtained by setting σ to be 0.001 in RBF kernel. This table shows similar prediction accuracy pattern to Table 1 but this time the highest prediction accuracy is achieved at $(\tau, \epsilon) = (0.01, 0.0001)$, too. The prediction accuracy 93.0% with $\sigma = 0.001$ is the greatest one among the whole data.

Table 4 is obtained by setting σ to be 0.0001 in RBF kernel. This table shows similar pattern of prediction accuracies as the previous tables but, we observe, the prediction accuracies are remarkably deteriorated.

Table 5 describes the total prediction accuracy of SVM classification with different σ values. The total prediction accuracy is the average of all the prediction accuracies at the points (τ, ϵ). The table shows that the total prediction accuracy gradually increases as σ decreases, until $\sigma = 0.001$. Roughly speaking, we have the best prediction accuracy when $\tau = 0.00001, \epsilon = 0.00001$ and $\sigma = 0.001$. Most of the total average prediction accuracies are over 90.0%. It means that

$\tau\backslash\epsilon$	0.1	0.01	0.001	0.0001	0.00001
0.10000	0.851266	0.844937	0.863924	0.860759	0.857595
0.01000	0.873418	0.844937	0.857595	0.867089	0.882911
0.00100	0.867089	0.873418	0.876582	0.879747	0.879747
0.00010	0.854430	0.876582	0.873418	**0.892405**	**0.892405**
0.00001	0.854430	0.879747	0.867089	**0.892405**	**0.892405**

TABLE 4. Prediction accuracy of SVM classification ($\sigma = 0.0001$).

σ	0.1	0.01	0.001	0.0001
Total Accuracy	0.902785	0.907848	0.915949	0.870253

TABLE 5. Total prediction accuracy of SVM classification with different σ.

SVM classification works well for solving status prediction. For $\sigma \geq 0.001$, the total average prediction accuracies of SAI preconditioner are slightly better than those of ILUT preconditioner [27]. It might be because the distribution of solving results of iterative solver with SAI preconditioner looks simpler than that of ILUT preconditioners.

5. Conclusion

There may be various parameter spaces which depend on the SAI algorithms. We choose two tolerance parameters of SAI and predict the solving status of a sparse linear system using PGMRES with SAI. It is difficult to predict all the possible areas in the parameter space that a given sparse linear system can be solved. We choose some sample points in the parameter space to predict the solving status of sparse linear systems at such sample points. Then we can have an idea of the outline of the area in the parameter space that the sparse linear system can be solved. We also analyzed in detail the area patterns in parameter space that can obtain high prediction accuracy in each prediction method. Generally speaking, we can get better prediction accuracy as the tolerance parameters τ, ϵ are smaller. It means that better prediction is achieved if we reduce the number of dropped entries for the original matrix and the preconditioner during computing the preconditioner. Our result can be added up to build a data mining techniques based intelligent preconditioner recommendation system for application scientists and engineers [20]. According to our experiment, the SAI preconditioner often succeeds to be computed even when the ILUT preconditioner for a sparse linear system fails, and vice versa. So ILUT and SAI preconditioners can complement each other.

References

1. M. W. Benson and P. O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.*, 22:127–140, 1982.
2. M. W. Benson, J. Krettmann, and M. Wright. Parallel algorithms for the solution of certain large sparse linear systems. *Int. J. Comput. Math*, 16:245–260, 1984.
3. E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. *SIAM J. Sci. Comput.*, 21(5):1804–1822, 2000.
4. E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19(3):995–1023, 1998.
5. J. D. F. Cosgrove, J. C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *Int. J. Comput. Math.*, 44:91–110, 1992.
6. T. Davis. University of Florida sparse matrix collection. *NA Digest*, 97(23), June 1997.
7. I.S. Duff and J.K. Reid A.M. Erisman. *Direct Methods for Sparse Matrices.* Clarendon Press, New York, NY, 1986.
8. M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18:838–853, 1997.
9. M. Grote and H. D. Simon. Parallel preconditioning and approximate inverse on the Connection machines. In R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold, and D. A. Reed, editors, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 519–523, Philadelphia, PA, 1993.
10. T. Joachims. Making large-scale svm learning practical. In C. Burges B. Schölkopf and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning.* MIT-Press, 1999.
11. N. Kang, E. S. Carlson, and J. Zhang. Parallel simulation of anisotropic diffusion with human brain dt-mri data. *Computers and Structures*, 82(28):2389–2399, 2004.
12. S. Karaa, C. C. Douglas, and J. Zhang. Preconditioned multigrid simulation of an axisymmetric laminar diffusion flame. *Math. Comput. Model.*, 38:269–279, 2003.
13. L. Y. Kolotilina. Explicit preconditioning of systems of linear algebraic equations with dense matrices. *J. Soviet Math.*, 43:2566–2573, 1988.
14. J. Lee, C. Lu, and J. Zhang. Performance of preconditioned Krylov iterative methods for solving hybrid integral equations in electromagnetics. *J. of Applied Comput. Electromagnetics Society*, 18:54–61, 2003.
15. *http://math.nist.gov/MatrixMarket.*
16. Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, New York, NY, 1996.
17. V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.
18. V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998.
19. K. Wang, S.-B. Kim, and J. Zhang. A comparative study on dynamic and static sparsity patterns in parallel sparse approximate inverse preconditioning. *Journal of Math. Model. Alg.*, 2(3):203–215, 2003.
20. S. Xu. *Study and Design of An Intelligent Preconditioner Recommendation System.* PhD thesis, Department of Computer Science, University of Kentucky, Lexington, KY, 2005.
21. S. Xu, E. Lee, and J. Zhang. Designing and building an intelligent preconditioner recommendation system. In *Abstracts of the 2003 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Napa, CA, 2003.
22. S. Xu, E. Lee, and J. Zhang. An interim analysis report on preconditioners and matrices. Technical Report 388-03, Department of Computer Science, University of Kentucky, Lexington, KY, 2003.

23. S. Xu and J. Zhang. A new data mining approach to predicting matrix condition numbers. *Commun. Inform. Systems*, 4(4):325–340, 2004.
24. S. Xu and J. Zhang. A data mining approach to matrix preconditioning problem. In *Proceedings of the Eighth Workshop on Mining Scientific and Engineering Datasets (MSD'05), in conjunction with the Fifth SIAM International Conference on Data Mining*, pages 49–57, Newport Beach, CA, 2005.
25. S. Xu and J. Zhang. Matrix condition number prediction with SVM regression and feature selection. In *Proceedings of the Fifth SIAM International Conference on Data Mining*, Newport Beach, CA, 2005.
26. S. Xu and J. Zhang. Solvability prediction of sparse matrices with matrix structure-based preconditioners. In *Abstracts of the 2005 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, Atlanta, GA, 2005.
27. S. Xu and J. Zhang. SVM classification for predicting sparse matrix solvability with parameterized matrix preconditioners. Technical Report 405-06, Department of Computer Science, University of Kentucky, Lexington, KY, 2006.
28. J. Zhang. Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications. *Comput. Methods Appl. Mech. Engrg.*, 189(3):825–840, 2000.
29. J. Zhang. Performance of ILU preconditioners for stationary 3D Navier-Stokes simulation and the matrix mining project. In *Proceedings of the 2001 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications*, pages 89–90, Tahoe City, CA, 2001.

Sang-Bae Kim received a Ph.D from Purdue University, USA, in 1993. He is a professor of Department of Mathematics, Hannam University, Korea. His research interests are numerical linear algebra, scientific computing and data mining.

Department of Mathematics, Hannam University, Daejon 306-791, Korea
`e-mail: sbk@hnu.kr`

Shuting Xu received a Ph.D. from University of Kentucky in 2005. She is a assistant professor of Department of Computer Information Systems , Virginia State University. Her research interests include preconditioning techniques for large scale matrix computation and data mining.

Department of Computer Information Systems, Virginia State University, Petersburg, VA 23806, USA
`e-mail: sxu@vsu.edu`

Jun Zhang received a Ph.D. from George Washington University in 1997. He is a Professor of Computer Science and Director of the Laboratory for High Performance Scientific Computing and Computer Simulation at the University of Kentucky. His research interests include large scale parallel and scientific computing, numerical simulation, iterative and preconditioning techniques for large scale matrix computation.

Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA
`e-mail: jzhang@cs.uky.edu`