Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 383 – 392

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Reusing Random Walks in Monte Carlo Methods for Linear Systems

Hao Ji[a], Yaohang Li[a]*

[a]*Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA*

Abstract

In this paper, we present an approach of reusing random walks in Monte Carlo methods for linear systems. The fundamental idea is, during the Monte Carlo sampling process, the random walks generated to estimate one unknown element can also be effectively reused to estimate the other unknowns in the solution vector. As a result, when the random walks are reused, a single random walk can contribute samples for estimations of multiple unknowns in the solution simultaneously while ensuring that the samples for the same unknown element are statistically independent. Consequently, the total number of random walk transition steps needed for estimating the overall solution vector is reduced, which improves the performance of the Monte Carlo algorithm. We apply this approach to the Monte Carlo algorithm in two linear algebra applications, including solving a system of linear equations and approximating the inversion of a matrix. Our computational results show that compared to the conventional implementations of Monte Carlo algorithms for linear systems without random walk reusing, our approach can significantly improve the performance of Monte Carlo sampling process by reducing the overall number of transition steps in random walks to obtain the entire solution within desired precision.

Keywords: Monte Carlo Method; Random Walk; Linear System; Inverse Matrix

1. Introduction

The Monte Carlo method is base on experimental mathematics using statistical sampling. Applying Monte Carlo methods to applications in linear systems is originally proposed by von Neumann and Ulam [1]. Considering a linear system of

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b}$$

where $\mathbf{H}$ is an M x M matrix, $\mathbf{b}$ is a given constant vector, and $\mathbf{x}$ is the unknown vector, the fundamental idea of the Monte Carlo solver is to construct Markov chains by generating random walks to statistically sample the underlying Neumann series

$$\mathbf{I} + \mathbf{H} + \mathbf{H}^2 + \mathbf{H}^3 + \ldots$$

of the linear system. If $\|\mathbf{H}\| < 1$, the Neumann series converge to $(\mathbf{I} - \mathbf{H})^{-1}$ and hence the mathematical expectation of the random walks starting from the ith row of matrix $\mathbf{H}$ equals to the solution x_i.

Other techniques have also been developed to improve the Monte Carlo algorithm for estimating the solutions of

* Corresponding author. Tel.: +1-757-683-6001; fax: +1-757-683-4900.
E-mail address: yaohang@cs.odu.edu.

1877-0509 © 2012 Published by Elsevier Ltd.
doi:10.1016/j.procs.2012.04.041

a linear system. Wasow [2] modified the scheme by von Neumann and Ulam by designing another unbiased estimator, which has been shown to have smaller variance under some special conditions. Halton [3] proposed a sequential Monte Carlo method to accelerate the Monte Carlo process by taking advantage of the rough estimate of the solution to transform the original linear system $\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b}$ to a new system $\mathbf{y} = \mathbf{H}\mathbf{y} + \mathbf{d}$, where $\|\mathbf{d}\| < \|\mathbf{b}\|$. Dimov et al. [4] developed an accelerating Monte Carlo scheme to control the convergence of the Monte Carlo algorithm for different unknown elements with different relaxation parameters, which can increase the efficiency of the random walk estimators. This iterative scheme is also used to approximate evaluation of the inverse matrix. Tan [5] studied the antithetic variates techniques for variance reduction in Monte Carlo linear solvers. Srinivasan and Aggarwal [6] used non-diagonal splitting to improve the Monte Carlo linear solvers. Moreover, for applications with large linear systems, Sabelfeld and Mozartova [7] designed a sparsified randomization algorithm by using a sparse, random matrix $\mathbf{G}$, which is an unbiased estimator of $\mathbf{H}$, to replace the original matrix $\mathbf{H}$ during the sampling process. Furthermore, Mascagni and Karaivanova [8] investigated the usage of quasirandom numbers in the Monte Carlo solver. Nevertheless, the fundamental mechanism of these Monte Carlo solvers, i.e., constructing Markov chains based on random walks to estimate solutions of the linear systems, remains the same.

The main drawback of the Monte Carlo methods is that they can only provide statistical estimates for a given linear system with a slow convergence rate of $O(N^{-1/2})$ [9], where N is the number of samples. Consequently, the Monte Carlo solvers are usually not as efficient as the modern deterministic direct or iterative solvers to obtain very precise solutions. However, in many important applications such as preconditioning [10], machine learning [11], and information retrieval [12], where the matrices are large while not very accurate solutions can be satisfactory, the Monte Carlo algorithms become attractive due to the fact that the Monte Carlo convergence rate is independent of the size of the matrix [9]. Moreover, similar to many Monte Carlo algorithms, most Monte Carlo methods for linear system are embarrassingly parallel [17], which are efficient and scalable on traditional parallel systems as well as a variety of emerging high performance computing platforms such as multi-core systems [13], the computational Grids [14], the Cloud computing architectures [15], and the General Purpose Graphics Processing Unit (GPGPU) clusters [16].

In the original Monte Carlo algorithm proposed by von Neumann and Ulam [1], a random walk can be used to estimate only one element in the unknown vector $\mathbf{x}$. Therefore, for applications interested in the general shape of $\mathbf{x}$, the original Monte Carlo algorithm is rather inefficient since separate random walks need to be generated for estimating different elements in $\mathbf{x}$. Reusing the random walks can improve the efficiency of Monte Carlo algorithms. Sbert et al. has shown that reusing the paths of random walks can reduce the cost of Monte Carlo algorithms in radiosity applications [19]. Moreover, Hammersley and Hamscomb [9] described an adjoint method for von Neumann and Ulam's Monte Carlo scheme, where a random walk can be used for estimate the overall solution $\mathbf{x}$. However, a potential problem is that the samples generated by the adjoint method are correlated, which may lead to biases in estimation of the solutions.

In this paper, we present a new approach of reusing random walks in the Monte Carlo algorithms for linear systems. In this reusing random walk scheme, during the Monte Carlo sampling process, a random walk can contribute samples for estimation of multiple unknown elements in $\mathbf{x}$ simultaneously and ensure that the samples for the same unknown element are independent in statistical sense. As a result, reusing random walks can significantly improve efficiency of Monte Carlo linear solver in estimating the entire solution $\mathbf{x}$ in a linear system. We use the iterative Monte Carlo scheme developed by Dimov et al. [4] as an example to demonstrate the effectiveness of our approach of reusing random walks. In addition to solving systems of linear equations, our approach of reusing random walks can be applied to Monte Carlo approximate evaluation of an inverse matrix, which is also shown in this paper.

The rest of the paper is organized as follows. We review the random walk method in the Monte Carlo linear system solver proposed by von Neumann and Ulam as well as the iterative scheme developed by Dimov et al. in Section 2. Sections 3 and 4 describe our reusing random walks approach and its applications in solving linear system and inverse matrix approximation, respectively. Section 5 presents our computational results. Finally, Section 6 concludes our summary and future research directions.

2. Random Walks and Monte Carlo Linear Solvers

Although different estimators are proposed in different Monte Carlo linear solvers, the fundamental mechanism

of the Monte Carlo algorithm is to construct Markov chains based on random walks to estimate the solutions of the linear systems.

The original Monte Carlo linear solver proposed by von Neumann and Ulam [1] employs a terminating random walk (terminating Markov chain). Considering a linear system of

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b},$$

where **H** is an M x M matrix and **b** is a given constant vector, an M x M transition probability matrix **P** is constructed by satisfying the following conditions:

$$P_{ij} \geq 0;$$

$$\sum_j P_{ij} \leq 1; \text{ and}$$

$$H_{ij} \neq 0 \rightarrow P_{ij} \neq 0.$$

The terminating probability vector **T** is defined as

$$T_i = 1 - \sum_j P_{ij}.$$

Then, a random walk γ starting at i_0 and terminating after k steps is defined as

$$\gamma = (i_0, i_1, \ldots, i_k),$$

where the integers $i_1 \ldots i_k$ are the row numbers visited during the random walk. The transition probability of random walk γ is

$$\mathrm{P}(i_{n+1} = j \mid i_n = i, n < k) = P_{ij}$$

and terminating probability is

$$\mathrm{P}(k = n \mid i_n = i, n - 1 < k) = T_i.$$

Also, define $X(\gamma)$ such that

$$X(\gamma) = \frac{H_{i_0 i_1} \ldots H_{i_{k-1} i_k} b_{i_k}}{P_{i_0 i_1} \ldots P_{i_{k-1} i_k} T_{i_k}}.$$

As a result, if the underlying Neumann series converges, $X(\gamma)$ is an unbiased estimator of component x_i in the unknown vector **x** where $i = i_0$.

Dimov et al. proposed an accelerating Monte Carlo scheme to improve the efficiency of the random walk estimator. The M x M transition probability matrix **Q** is defined as

$$Q_{ij} = \frac{|H_{ij}|}{\sum_j |H_{ij}|}.$$

Instead of defining an explicit terminating probability, for a random walk γ starting at i_0,

$$\gamma = (i_0, i_1, \ldots),$$

a weight quantity W_n is calculated at each transition step by

$$W_0 = 1,\ W_1 = W_0 * \frac{H_{i_0 i_1}}{Q_{i_0 i_1}}, \ldots, W_n = W_{n-1} * \frac{H_{i_{n-1} i_n}}{Q_{i_{n-1} i_n}}, \ldots$$

The random walk γ will be terminated at the kth step when $|W_k| \leq \varepsilon$ for the first time, where ε is a given tolerance. Similarly, if the underlying Neumann series converges,

$$Y(\gamma) = \sum_{n=0}^{k} W_n b_{i_n}$$

is an unbiased estimator for x_i where $i = i_0$.

3. Reusing Random Walks in Monte Carlo Solver for Linear Systems

3.1. Analysis of Random Walk Reusing

In the original Monte Carlo linear solver by von Neumann and Ulam or the alternative scheme by Dimov et al., a random walk is used to generate a sample for a single component x_i in the unknown vector **x** by initiating the random walk at the *i*th row in the transition matrix. Generally, Monte Carlo algorithms require sufficient number of independent samples to obtain a reasonably accurate estimation of a solution. Assuming that averagely each element in the unknown vector **x** requires N independent samples to achieve certain accuracy, finding the overall solution of **x** with M unknown components demands $M * N$ random walks. Figure 1 illustrates 50 independent random walks generated for producing 50 samples to compute every unknown element in a 6 x 6 system of linear equations without being reused. This is rather costly particularly when the linear system is large.

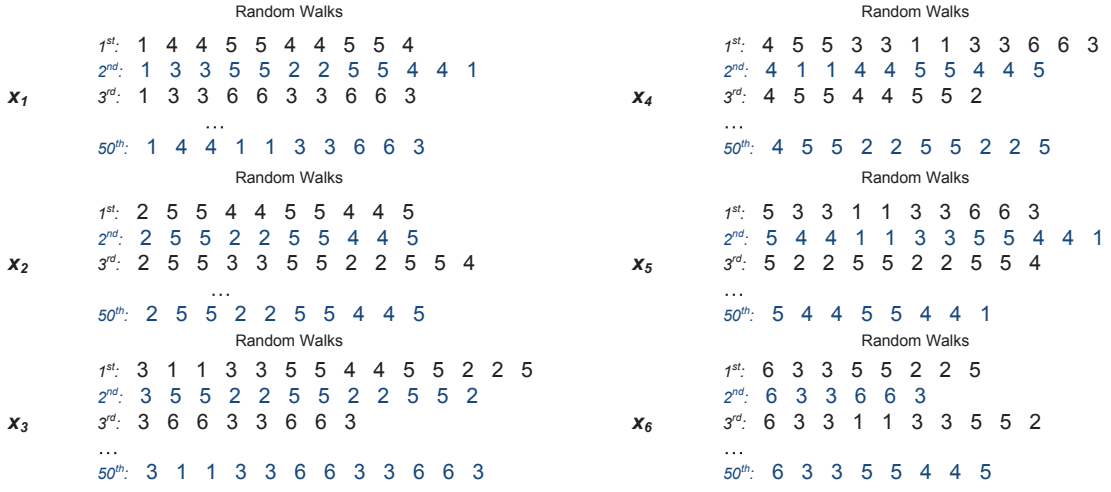

Figure 1: 50 Independent Random Walks Generated for Estimating Each Unknown Element in a 6 x 6 System of Linear Equations without Being Reused

Our further analysis finds that Monte Carlo algorithms require statistical independence of samples when estimating the same element in the unknown vector **x**; however, such independence is not necessary among samples for different elements. In other words, the samples for one element can be correlated with those for a different element. As a result, a random walk, or at least part of it, for generating samples for one element can be reused for estimating another different one.

Consider the 1st random walk for estimating x_5 shown in Figure 1

$5 \rightarrow 3 \rightarrow 3 \rightarrow 1 \rightarrow 1 \rightarrow 3 \rightarrow 3 \rightarrow 6 \rightarrow 6 \rightarrow 3$.

After transition from 5 to 3, the partial random walk $3 \rightarrow 3 \rightarrow 1 \rightarrow 1 \rightarrow 3 \rightarrow 3 \rightarrow 6 \rightarrow 6 \rightarrow 3$ can be regarded as part of a random walk starting at 3 and thus it can be reused to estimate element x_3. Similarly, after the random walk reach 1, the partial random walk $1 \rightarrow 1 \rightarrow 3 \rightarrow 3 \rightarrow 6 \rightarrow 6 \rightarrow 3$ can also be reused to estimate element x_1 and so on. Reusing the random walks can reduce the total number random walk steps needed for estimating the overall solution vector **x** and therefore improve the performance of the Monte Carlo solver.

3.2. Monte Carlo Linear Solver with Random Walk Reusing

We use the iterative Monte Carlo scheme developed by Dimov et al. [4] as an example to illustrate our implementation of reusing random walks in Monte Carlo linear solvers. Figure 2 shows the pseudocode of the Monte Carlo linear solver with random walk reusing. The main difference between the conventional scheme and the reusing random walk scheme is that a reusing random walk will not terminate until the current random walk no longer contributes to any unknown elements demanding more samples. During the sampling process, a contribution set S is used to keep track of the indices of the unknown elements in the solution vector being contributed by the current random walk and an array *flag* is used to indicate if there is sufficient number of samples for each unknown element. For each unknown element x_i, a data structure is used to store the current weight quantity (W) and the current sample value (X). A random walk starts from the index of a randomly selected unknown element requiring

more samples. When convergence of the random walk on an unknown element x_i is observed, a sample for x_i is generated and index i is removed from contribution set S, i.e., the current random walk no longer contribute to x_i. Afterward, the statistical variance of x_i is estimated. If the statistical error is less than the statistical error bound σ, the corresponding value in the *flag* array is set, indicating that x_i does not need more samples. If contribution set S is empty indicating that the current random walk no longer contributes to any unknown elements in need of more samples, the random walk will stop. Then, a new random walk is spawn starting from the index of one of the unknown elements still requiring more samples. This process repeats until all unknown elements in solution vector **x** obtain sufficient number of random samples to achieve desired statistical accuracy.

```
Input:
    Coefficient matrix H, constant vector b, tolerance ε, and statistical error bound σ
Preprocessing:
    Calculate the probability matrix P and the row sum vector rsum where rsum(i)= ∑_{j=1}^{M} |H_ij|;
Monte Carlo algorithm with random walks reusing:
    While R = {k | flag(k) == 0 and k ∈ {1, 2, ..., M}} is not empty
      Begin    // starting a new random walk
        Select a starting state s ∈ R
        Set S = {s}
        Set x_s.W = 1 and x_s.X = b_s
          While the set S is not empty
            Begin
              Generate an uniformly distributed random number θ ∈ [0,1)
              Locate index j satisfying ∑_{t=1}^{j-1} P_st ≤ θ ≤ ∑_{t=1}^{j} P_st
              For each element x_i whose index i ∈ S // contribute to multiple elements
                  Begin
                    Calculate x_i.W = x_i.W * sign(H_sj) * rsum(s)
                    Calculate x_i.X = x_i.X + x_i.W * b_j
                    If |x_i.W| < ε, then // convergence
                      Begin
                        Generate x_i.X as a sample for x_i
                        Delete s from the set S
                        Calculate variance of x_i
                        If statistical error of x_i < σ then
                            Begin // random walk no longer contribute to x_i
                              Set flag(i) = 1
                            End
                      End
                  End
              Set s = j // update the random walk state
              If flag(s) == 0 and s ∉ S, then
                  Begin
                    Add s into the set S      // contribute to a new element
                    Set x_s.W = 1 and x_s.X = b_s
                  End
            End
      End
```

Figure 2: Pseudocode of Monte Carlo Linear Solver with Reusing Random Walks

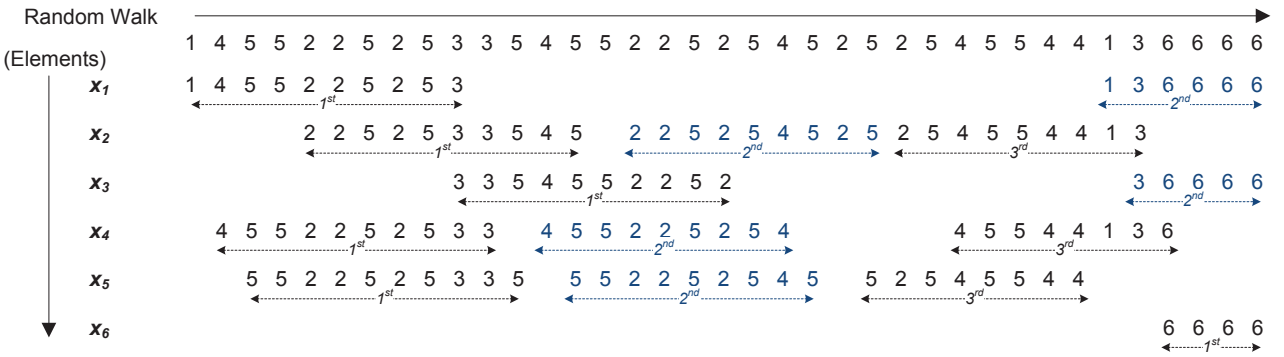

Figure 3: Example of a Random Walk being Reused for Estimating 6 Unknown Elements in a 6x6 Linear System

Figure 3 illustrates an example of a hypothetical random walk starting from 1 being reused for estimating 6 unknown elements in a 6 x 6 linear system. The reusing random walk implementation shown in Figure 2 ensures that the random samples generated for the same element are statistically independent without overlapping trajectories.

4. Reusing Random Walks for Approximating an Inverse Matrix

One can rewrite the linear system $\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b}$ into a more common form

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A} = \mathbf{I} - \mathbf{H}$ is another M x M matrix. Setting

$$b_i = (0, \quad 0, \quad ..., \quad \underbrace{1}_{i}, \quad ..., \quad 0)^T,$$

solution vector $\mathbf{x_i} = (x_{1i}, x_{2i}, \cdots, x_{Mi})^T$ of $\mathbf{A}\mathbf{x_i} = \mathbf{b_i}$ is the ith column of the inverse matrix $\mathbf{A}^{-1}$. Therefore, repeatedly using Monte Carlo linear solver to evaluate the each column of the inverse matrix of $\mathbf{A}^{-1}$, one can have an approximate estimation of the inverse matrix $\mathbf{A}^{-1}$.

Reusing random walks can be also extended to the application of approximating an inverse matrix. Figure 4 depicts a scheme of reusing a random walk in approximating the inverse matrix of a 6 x 6 matrix. A single random walk step can contribute to the evaluations of the elements in an entire row while statistical estimations of multiple rows in $\mathbf{A}^{-1}$ can share the same random walk.

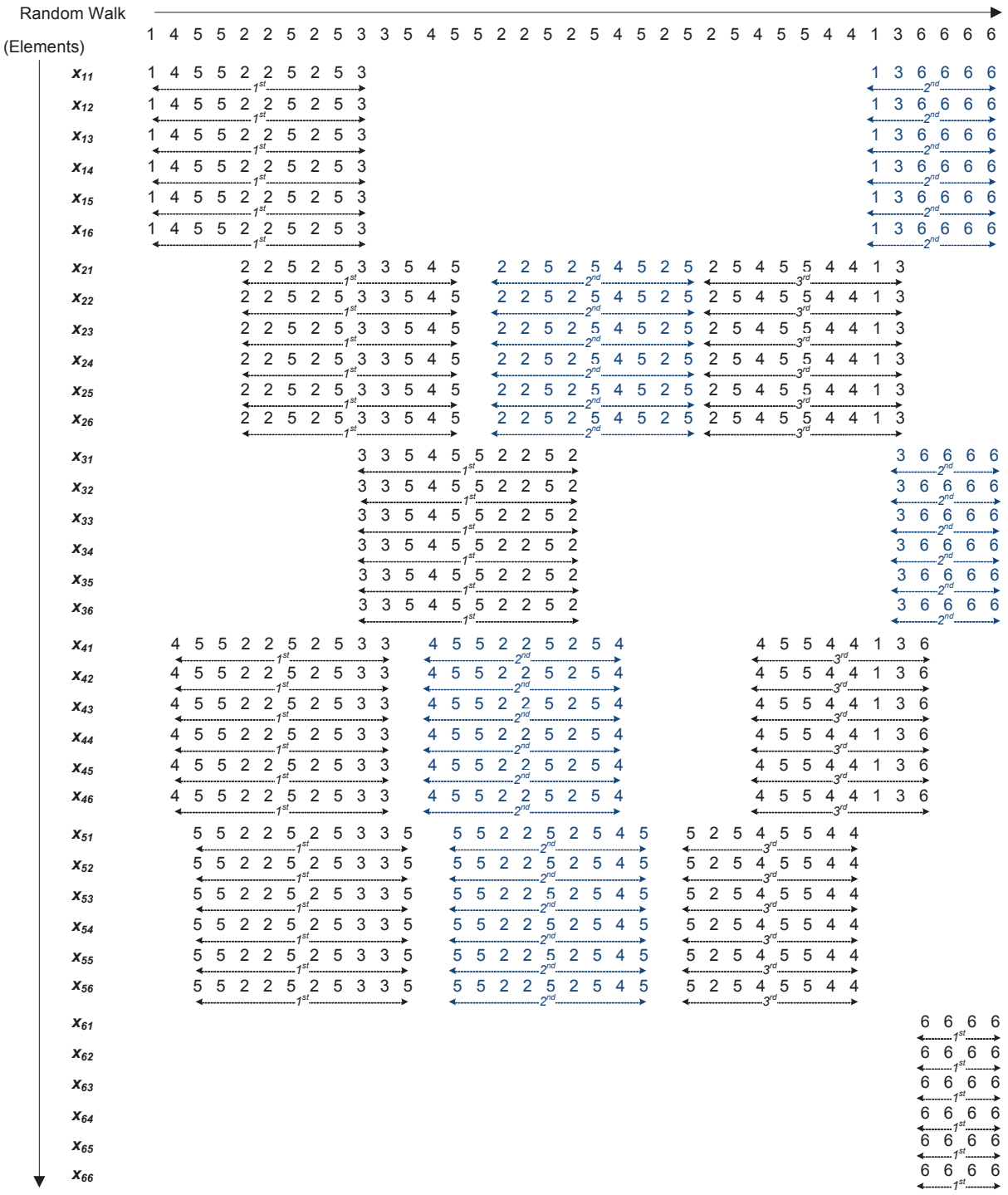

Figure 4: Reusing a Random Walk in Approximating the Inverse Matrix of a 6 x 6 Matrix

5. Computational Results

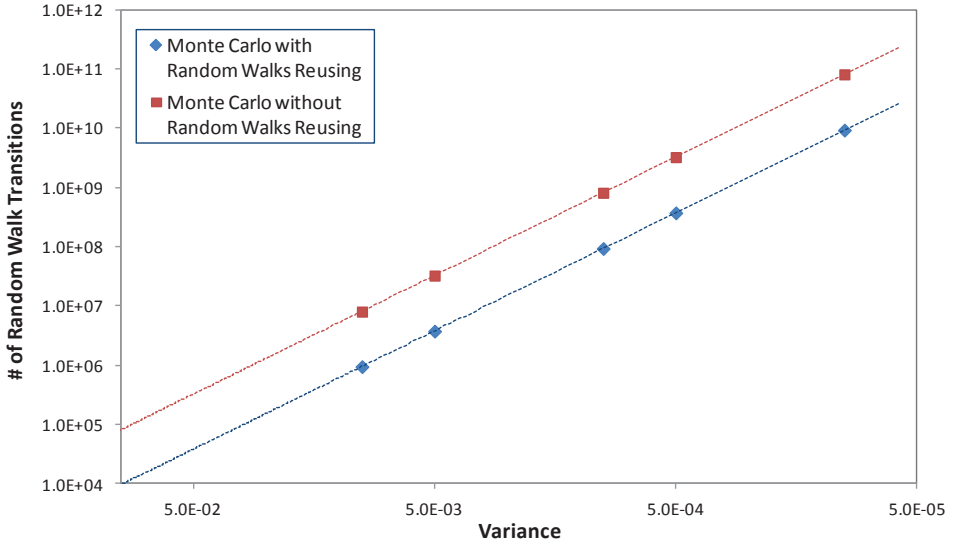

Figure 5: Performance comparison of Monte Carlo linear solver with random walks reusing and without random walks reusing under different precisions. The linear system uses "mesh2e1" as coefficient matrix. The computational results are based on the averages of 20 runs.

We use a strictly diagonally dominant matrix "mesh2e1" obtained from the UFL sparse matrix collection [18] as the coefficient matrix for a linear system in this computational experiment. "mesh2e1" is a 306 x 306 matrix with 2,018 nonzero elements. Figure 5 compares the numbers of random walk transitions in Monte Carlo linear solver to obtain the entire solution vector in different precisions with random walks reusing and the one without random walks reusing. One can find that our random walk reusing approach can reduce the number of random walk transitions by approximately 5.8 times. A random walk transition is a relatively costly operation in the Monte Carlo linear solvers. Considering a row in the coefficient matrix with K nonzero elements, $\lfloor \log(K) \rfloor$ comparison operations are needed to find the next state by comparing the transition probabilities if binary search is used.

Figure 6 compares the numbers of random walk transitions in the Monte Carlo algorithm of approximating the inverse matrix of "mesh2e1" with different precisions with random walks reusing and the one without random walks reusing. Reusing random walks in inverse matrix estimation is even more effective than its implementation in Monte Carlo linear solver. The total number of transition is reduced by about 56.9 times when the random walks are reused in calculating the inverse matrix of "mesh2e1." This is due to the fact that in inverse matrix approximation using Monte Carlo algorithms, a random walk can contribute to more elements – a random walk can be potentially reused for estimating different rows and a transition in the random walk can also be used to evaluate the elements in the entire rows that it is contributing to.

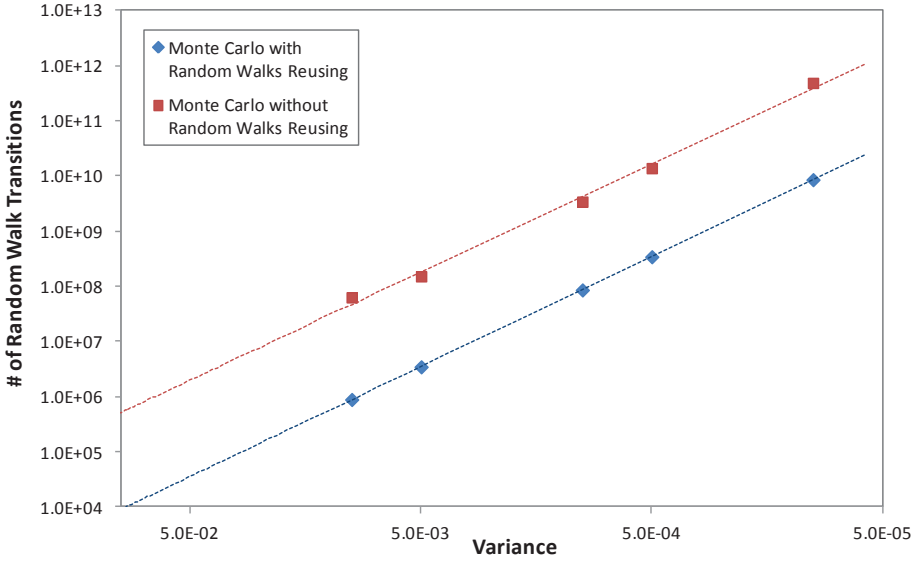

Figure 6: Performance Comparison of Monte Carlo Algorithms of Approximating the Inverse Matrix "mesh2e1" with Reusing Random Walks and without Reusing Random Walks in different precisions. The computational results are based on the averages of 20 runs.

6. Conclusions and Future Research Directions

In this paper, we present an approach of reusing random walks in Monte Carlo algorithms for linear systems to improve their efficiency in obtaining the entire solution. When the random walks are shared, a single random walk can contribute to estimations of multiple unknowns in the solution vector simultaneously. We apply the reusing random walk techniques to the Monte Carlo algorithms of solving a linear system as well as approximating a matrix's inversion. Our computational results show that compared to the conventional implementations of the Monte Carlo algorithms, our reusing random walks approach can significantly reduce the overall number of random walk transition steps during the Monte Carlo sampling process to obtain solutions within desired precision.

We use the iterative scheme by Dimov et al. [4] as an example to illustrate our implementation of reusing random walk in this paper. In general, the random walk reusing approach is applicable to other Monte Carlo estimators as well.

One of the main advantages of the Monte Carlo methods is their high parallel efficiency. Our future work will focus on developing efficient parallel Monte Carlo algorithms for linear algebra applications with reusing random walks on traditional and emerging parallel computing platforms.

Acknowledgements

We would like to thank Dr. Michael Mascagni for his valuable discussions on reusing random walk paths. Yaohang Li acknowledges support from NSF grant 1066471 and ODU 2011 SEECR grant.

References

[1] G. E. Forsythe, R. A. Leibler, "Matrix inversion by a Monte Carlo method," Mathematical Tables and Other

Aids to Computation, **4**: 127-129, 1950.
[2] W. Wasow, "A note on the inversion of matrices by random walks," Mathematical Tables and Other Aids to Computation, **6**: 78-81, 1952.
[3] J. H. Halton, "Sequential Monte Carlo techniques for the solution of linear systems," Journal of Scientific Computing, **9**: 213-257, 1994.
[4] I. T. Dimov, T. T. Dimov, T. V. Gurov, "A new iterative Monte Carlo Approach for Inverse Matrix Problem," Journal of Computational and Applied Mathematics, **92**: 15-35, 1998.
[5] C. J. K. Tan, "Antithetic Monte Carlo Linear Solver," Proceedings of the International Conference on Computational Science (ICCS02), 2002.
[6] A. Srinivasan, V. Aggarwal, "Improved Monte Carlo linear solvers through non-diagonal splitting," Proceedings of the International Conference on Computational Science and its Applications (ICCSA03), 2003.
[7] K. Sabelfeld, N. Mozartova, "Sparsified randomization algorithms for large systems of linear equations and a new version of the random walk on boundary method," Monte Carlo Methods and Applications, **15**(3): 257-284, 2009.
[8] M. Mascagni, A. Karaivanova, "A Parallel Quasi-Monte Carlo Method for Solving Systems of Linear Equations," Proceedings of International Conference on Computational Science (ICCS02), 2002.
[9] J. M. Hammersley, D. C. Handscomb, "Monte Carlo Methods," Chapman and Hall, 1964.
[10] H. Ji, D. Nguyen, Y. Liang, Y. Li, "Incomplete Jacobi rotation and its applications in large, sparse, symmetric and non-symmetric linear systems," submitted to Applied Mathematics and Computation, 2012.
[11] A. Talwalkar, S. Kumar, H. A. Rowley, "Large-scale manifold learning," Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR08), 2008.
[12] K. Bryan, T. Leise, "The $25,000,000,000 eigenvector: the linear algebra behind Google," SIAM Review, **48**(3): 569-581, 2006.
[13] S. Tomov, R. Nath, H. Ltaief, J. Dongarra, "Dense linear algebra solvers for multicore with GPU accelerator," Proceedings of IEEE International Symposium on Parallel and Distributed Processing (IPDPS10), 2010.
[14] I. Foster, C. Kesselman, S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," International Journal of High Performance Computing Applications, **15**(3): 200-222, 2001.
[15] C. Vecchiola, S. Pandey, R. Buyya, "High-performance cloud computing, a view of scientific applications," Proceedings of 10th International Symposium on Pervasive Systems, Algorithms, and Networks, (ISPAN09), 2009.
[16] J. Krüger, R. Westermann, "A GPU framework for solving systems of linear equations," GPU Gems 2, Chapter 44, 2009.
[17] Y. Li, M. Mascagni, "Analysis of Large-scale Grid-based Monte Carlo Applications," International Journal of High Performance Computing Applications (IJHPCA), **17**(4): 369-382, 2003.
[18] UFL sparse matrix collection. http://www.cise.ufl.edu/research/sparse/matrices/index.html, 2012.
[19] M. Sbert, P. Bekaert, J. Halton, "Reusing paths in radiosity and global illumination," Monte Carlo Methods and Applications, **10**: 575-585, 2004.