Modeling the Locality in Graph Traversals

Liang Yuan[*†‡], Chen Ding[§], Daniel Štefankovič[§] and Yunquan Zhang[*†]
[*]Lab. of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences
[†]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[‡]Graduate University of Chinese Academy of Sciences
[§]Computer Science Department, University of Rochester
yl.iscas@gmail.com zyq@mail.rdcps.ac.cn {cding,stefanko}@cs.rochester.edu

***Abstract*—An increasing number of applications in physical and social sciences require the analysis of large graphs. The efficiency of these programs strongly depends on their memory usage especially the locality of graph data access. Intuitively, the locality in computation should reflect the locality in graph topology. Existing locality models, however, operate either at program level for regular loops and arrays or at trace level for arbitrary access streams. They are not sufficient to characterize the relation between locality and connectivity.**

This paper presents a new metrics called the vertex distance and uses it to model the locality in breadth-first graph traversal (BFS). It shows three models that use the average node degree and the edge distribution to predict the number of BFS levels and the reuse distance distribution of BFS. Finally, it evaluates the new models using random and non-random graphs.

***Keywords*-locality; graph traversals; vertex distance; reuse distance;**

I. Introduction

Graphs are widely used in many fields such as n-body simulation, molecular biology, and Internet and social networks. Many problems in computer science can be formulated as graph problems. Among the graph algorithms, Breadth-First Search (BFS) is a basic routine. To represent irregular computation, Breadth-First Search has been proposed as a kernel in the *Graph500* benchmark for evaluating parallel computing system.

BFS and other algorithms have many variants designed to improve their performance on different platforms including shared-memory systems [1]–[7], distributed memory systems [8]–[11], GPU [12]–[14] and external-memory [15], [16]. All these methods try to improve the locality of data access. The effect, however, depends highly on the input graph, not just on the size but also its topology. For the same size, a graph may have a different structure and different locality. To select the right algorithm and its parameters, it would help if we can model the locality of an input graph based on high-level characteristics of the graph.

A machine-independent metric of locality is *reuse distance*. For each memory access in a program execution, the reuse distance is the number of distinct data elements accessed between this and the previous access to the same data. The distribution of all reuse distances is called the *locality signature*. It shows the average locality of the execution. The relation between the locality signature and the miss rate has been well established, both for direct mapped and set-associative cache [17]–[19].

In this paper, we propose a new concept called vertex distance. It counts the number of times other graph nodes are accessed between two consecutive visits of the same graph node. Unlike reuse distance, vertex distance is a measure of time rather than data volume. It is more amenable to traditional analysis of time and space complexity and can lead to an estimate of locality, which is a measure of active data usage.

We show that vertex distance can be used to infer reuse distance in BFS for two types of graphs. The first is a random graph. The second is a R-MAT (Recursive MATrix) graph [20]. R-MAT is a recursive graph generator capable of producing both uniform and non-uniform graphs.

Based on the vertex distance, we describe three models. The first is a probabilistic model for the random graph. It can predict the locality very precisely because there is no small-world property in random graphs [21]. The second model extends the first by considering the graph topology. We use the two models to predict the BFS tree, i.e. the number of levels in the BFS tree and the number of vertices in each level, and the locality of BFS, i.e. the reuse-distance signature. Finally, the last model predicts the distribution of vertex distances in random graphs, removing the need of profiling in locality analysis. These models reveal the effect of graph characteristics on the locality of BFS graph traversal. We believe that a similar approach can be used to model the locality in other types of graph traversal.

Previous locality models are general rather than graph specific. They use abstractions such as data access traces [19], loops and arrays [22], or data access probabilities [23]. Vertex distance builds on access trace based models but over a graph-specific abstraction. This paper shows the relation between vertex distance and reuse distance and how the relation is affected by the graph topology. These problems cannot be addressed directly using the general models.

The rest of this paper is structured as follows. Section 2 provides the background on BFS and locality analysis. Section 3 describes the vertex distance and the three locality models. Section 4 presents experimental results. Finally, Section 5 summarizes the strength and the limitations and points to future work.

Algorithm 1 Record Vertex Distance

Input: Graph G with n nodes, BFS $root$

Output: $VDCnt[1 \dots n], Level[1 \dots n]$

```
1:  $Level[1 \dots n] \leftarrow \infty$, $VDCnt[1 \dots n] \leftarrow 0$
2:  $Level[root] \leftarrow 0$, $CurVertex \leftarrow 0$
3:  $LastVertex[root] \leftarrow CurVertex$
4:  $Enqueue(BFSQueue, root)$
5:  while $!Empty(BFSQueue)$ do
6:    $i \leftarrow Dequeue(BFSQueue)$
7:    for $j \in Neighbor(i)$ do
8:      if $Level[j] == \infty$ then
9:        $Level[j] \leftarrow Level[i] + 1$
10:       $Enqueue(BFSQueue, j)$
11:     else
12:       $VD \leftarrow CurVertex - LastVertex[j]$
13:       $VDCnt[VD] \leftarrow VDCnt[VD] + 1$
14:     end if
15:     $LastVertex[j] \leftarrow CurVertex$
16:   end for
17:   $CurVertex \leftarrow CurVertex + 1$
18: end while
```

TABLE I
R-MAT GRAPH PARAMETERS AND SIZES

a	b	c	d	input ef	n	e	real ef
1	1	1	1	13	16384	425656	25.9
4	1	1	4	14	16384	423024	25.8
7	1	1	7	17	16384	411046	25.0

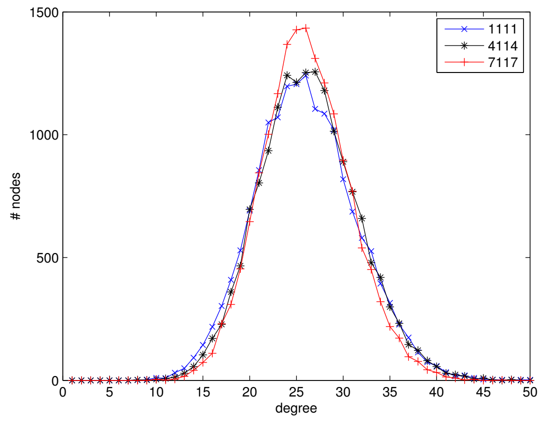


Fig. 1. The edge degree distributions of the three R-MAT graphs

II. Background

A. Breadth-first Search

Breadth-First Search (BFS) is one of the basic methods to traversal a graph. Given a graph $G = (V, E)$ $(n = |V|, m = |E|)$ and a vertex $root$, BFS traverses all the nodes by the order of $distance$ (or *Level*) which is the number of edges in the shortest path from the root. Thus, each vertex will be labeled with a *Level* number. The BFS algorithm will process nodes of a smaller level number before those of a larger level number. A simple queue-based implementation is shown in Algorithm 1 (for now ignore line 12-16 which will be used for the model).

In BFS, only the BFS tree (implemented by the BFS *Level* array in Algorithm 1) is reused. To measure locality, we record the reuse distance of the accesses of the BFS array. The total number of accesses equals to the number of edges m.

B. Graph Generation

There are two main random graph generators: *G(n,p)* [24] and *G(n,m)* [25]. In *G(n,p)*, n is the number of nodes in the graph, and p the probability that a pair of nodes are connected. In *G(n,m)*, m is the number of edges in the graph. If $m = p \cdot n \cdot (n-1)/2$, the two models have similar expectations in graph topology. We consider only the *G(n,p)* model. Random graphs are uniform in how they are generated.

To study non-uniform graphs, we use R-MAT [20] from the Graph500 package. The input to R-MAT is the size of a graph n, the average number of edges per node r, and a tuple of four parameters a, b, c, d, representing a 2×2 matrix. R-MAT first creates n graph nodes. The edges are given by the $n \times n$ adjacency matrix. An edge is generated by recursively using the parameter matrix to choose among the four quadrants of the adjacency matrix and eventually activating a single cell. For undirected graphs, both the parameter matrix and the adjacency matrix are symmetrical.

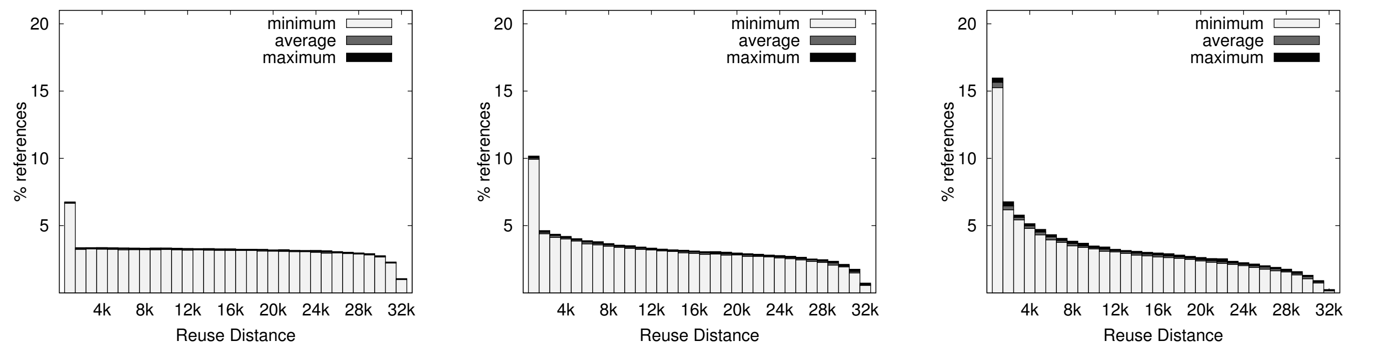
C. The Locality of Basic BFS

We use an example to show the effect of graph topology on locality. Prior work, e.g. a model built by Zhong et al. [19], used the input size to predict locality. For graphs, however, graphs of the same size may exhibit different locality.

As described in Section II-B, we use R-MAT to generate three graphs of different structures. Table I lists the parameters and the structure information of the three graphs, one in each row. The four input parameters are given in the first four columns.

We refer to the three graphs by these four parameters, namely, *1111*, *4114* and *7117*. *1111* is a uniform random graph where every two vertices in the graph has the same probability to be connected by an edge. *4114* and *7117* are uneven in that in every sub-graph, half of the graph has 4 and 7 times as many edges as the other half, respectively.

The next four columns of Table I show the input and real edge factors and the number of nodes and edges. An edge factor, represented as ef in the table, shows how many neighbors each node has. In R-MAT generation, more than one edge may be generated between two vertices. These duplicates are then removed. Uneven graphs contain more duplicates. To equalize the size of the three graphs, we use a larger input edge factor for the uneven graphs so the real edge factor is roughly the same in the generated graphs. The three graphs have the same number of vertices and an almost identical number of



(a) 1111 (b) 4114 (c) 7117

Fig. 2. Measured BFS Reuse Distance for R-MAT $(n = 16,384, ef = 16)$. The range in each bin reflects the difference when using different root nodes in BFS.

edges.

Intuitively we can expect that the two non-uniform graphs, *4114* and *7117*, have better locality than the uniform graph, *1111*. This is confirmed by the reuse distance measurement shown in Figure 2. BFS on *1111* has more long-distance reuses than on *4114* and *7117*. Thus the graph structure is an important factor in determining the locality of BFS. We expect the same conclusion for other graph algorithms. Merely the size of a graph—the number of nodes and the number of edges—will not be sufficient to characterize locality.

Three more issues are worth noting. The first is the effect of topology on the number of neighbors. Figure 1 shows the degree distribution of the three graphs. The most uneven graph, *7117*, has fewer nodes that have too many or too few neighbors, compared to the even graph, *1111*. The second is the starting point in BFS. Any node can be chosen as the root for a BFS traversal. We have randomly selected the starting points and tested a number of executions. The range of results is shown in Figure 2 by the minimum and maximum in each bar of the reuse-distance signature. The small difference shows that the starting point has little or no impact on locality. The reuse distance of BFS started from any root is similar to each other. The last issue is the numbering of graph nodes. By conducting experiments similar to those testing the effect of the starting point, we found that the numbering does not affect locality. The reuse distance of BFS with different vertex numbering is similar to each other.

D. Improved BFS Algorithms

BFS and other graph algorithms are the subject of many studies that focus on improving performance, especially memory and communication performance. We next classify these algorithms based on the target platform.

BFS on Shared Memory System. Bader and Madduri [1] presented fast multithreaded algorithms for BFS and st-connectivity on the MTA system. Zhang and Hansen [2] used a *layer synchronization* method for parallelizing BFS. This method is also used by many improved algorithms. Xia and Prasanna [3] proposed a topologically adaptive parallel algorithm which estimates the scalability of each BFS level and adjusts the number of threads according to the estimated value of scalability. You et al. [4] explored alternatives to atomics in creating the stack of newly-visited vertices. The new algorithm aggregates all the edges out of the current level, sorts them by the destination vertex and then performs the visited check. This will incur a much higher computational cost for sorting.

Leiserson and Schardl [6] replaced the shared BFS queue with a new bag data structure which is an unordered set supporting $O(\log n)$ time union and split. Agarwal et al. [5] designed a parallel BFS algorithm minimizing the negative effects of the cache-coherency protocol between processor sockets. Their algorithm utilizes the intra-level parallelism. In each BFS level, every core processes the vertices which belong to it in the current queue, updates the adjacent vertices which belong to itself, and sends accessed adjacent vertices to their owners who are in charge of updating the receive vertices. They use a bitmap array to mark the visited vertices, which reduces the working set size, and check the bitmap before locking it, which reduces the frequency of the potentially expensive atomic operation.

Pearce et al. [7] studied graph processing in a semi-external memory scenario, which has enough main memory to store the vertices but not edges. They proposed multithreaded asynchronous algorithms for SSSP, BFS and connected components. The idea is in the middle point between ordered algorithm and unordered algorithms [26]. It can provide more parallelism but requires more work.

Distributed Memory System. Because checking whether a remote node has been visitied needs inter-process communication, which incurs longer latency, the common approach is to gather all edges corresponding to remote nodes and send them to their owner processors at the end of each level. This requires an all-to-all communication.

Yoo et al. [11] proposed a distributed level-synchronized BFS algorithm using 1D partitioning and 2D partitioning. The 2D partitioning BFS algorithm replaces the expensive all-to-all communication step by a process-row and a process-column communication at each level. Buluc and Madduri [9] used SpMV to compute the BFS level and improved the 2D partition method [11] by using 2D vector distribution, which improves the load balance by reducing the next level queue. To reduce the storage of the row_ptr array, they used the doubly-compressed sparse column (DCSC) introduced for hypersparse matrices designed by Buluc and Gilbert [27].

BFS on External Memory. Munagala and Ranade [16] improved the I/O complexity of BFS (MR_BFS) to $O(n + \text{sort}(n+m))$ by generating the multiset of neighbor vertices of nodes in the current level, sorting and scanning it, deleting duplicates elements which are in the two most recently completed BFS levels. Mehlhorn and Meyer [15] proposed an improved algorithm (MM_BFS) based on MR_BFS by splitting the graph into subgraphs of a low diameter. The complexity is $O(\sqrt{n \cdot \text{scan}(n+m)} + \text{sort}(n+m))$. Ajwani et al. [28] experimentally studied MR_BFS and MM_BFS and showed that MR_BFS performs better on small-diameter random graphs but MM_BFS outperforms MR_BFS for large diameter sparse graphs.

Graph Topology Aware. Idwan and Etaiwi [10] use hMETIS to partition a graph into subgraphs which can be stored in memory. The algorithm processes BFS inside each subgraphs to reduce the number of I/O operations.

Muntes-Mulero et al. [8] proposed two graph partitioning methods to reduce the inter-node communication of the distributed BFS algorithm. In their BFS algorithm each node keeps information about the vertices in other partitions connected to vertices in the current partition and sends these vertcies to their owners only once. They proposed two graph partitioning strategies for BFS in distributed memory system. Their methods reduce the number of boundary vertices which have at least one cut edge. The improved φ-aware DBFS algorithm based on [11] doesn't send a message for each cut edge, but for distinct nodes connected to a cut edge.

Cong and Sbaraglia [29] studied the locality of three Minimum Spanning Tree algorithms. For the Prim algorithm, the heap operations are the main analysis target. Their experimental results show the Prim algorithm has the best locality (measured by reuse distance) and a consistent locality curve over different sparse graphs, and Kruskal has the worst locality.

III. Modeling Graph Locality with Vertex Distance

In this section, we define vertex distance and use it to analyze random graphs in the first and third models and R-MAT graphs in the second model. The first two models rely on directly measuring the vertex distance, while the third model derives the vertex distance.

A. The Vertex Distance

Graph data is accessed differently than array data. Algorithms such as BFS, Dijkstra shortest path, and Prim's minimum spanning tree share a common scheme: The access expands from the set of already accessed nodes outward into the rest of the graph. The expansion is done by following edges of each node. Since more than one node may be expanded at a given moment, the ready nodes are ordered linearly. An example of the ordering is the queue used in BFS. As nodes are queued, $v_1, v_2, \ldots, v_n$, the edges of v_1 are processed first, followed by the edges of v_2 and so on. We call the linear order in which graph nodes are first reached by an algorithm the *spanning sequence* of the algorithm.

The spanning sequence can be viewed as an array. Each node is added to the sequence once and only once. We use the array index of the nodes to compute a distance metric for their access. We define the *vertex distance* for each access as follows: In BFS, if a node x has r neighbors, it will be accessed r times, each time from a different neighbor. Sort the r neighbors by their index in the spanning seqence. Let neighbors r_i, r_{i+1} be adjacent in the sorted neighbor sequence. The difference in their indices is the vertex distance of the x access due to neighbor r_{i+1}.

The vertex distance is defined for each access. BFS follows every edge and accesses the end node at each edge. The vertex distance is effectively defined for each edge. For a node with r edges connecting to r neighbors, the vertex distance is infinite when x is accessed through the first neighbor in the spanning sequence. The set of remaining $r-1$ vertex distances show the distribution of the neighbors in the spanning sequence.

The histogram of vertex distances is the *vertex distance signature*, which we denote as $VD(n)$ as we use $RD(n)$ to denote the reuse-distance signature. For a graph with n nodes and e edges, there are $e-n$ finite vertex and reuse distances.

Algorithm 1 shows how to record the vertex distances in BFS, specifically, the reuse of the *Level* array in Line 8-9. It keeps a record called *LastVertex* to store the index to the spanning sequence, *BFSQueue*. When a vertex v is taken from *BFSQueue*, all neighbor vertices are accessed and the *LastVertex* array is updated to the index of v for each neighbor.

For example, when a vertex v is first accessed (line 9-10) from neighbor s_1, its *Level* is assigned, and v is added to the BFS queue. *LastVertex* is assigned the index of s_1. When v is accessed again from neighbor s_2, it is a temporal reuse. The vertex distance is the index of s_2 minus the index of s_1 stored in *LastVertex*. The vertex distance is recorded (line 12-13) and the value in *LastVertex* updated.

A vertex distance is a period of time, that is, the number of computation steps before a node is accessed again. The metric helps to connect the complexity analysis, where the time cost is the output, with the locality analysis, where the time distance is the input.

B. The Percolation Model for Random Graphs

In this first model, we show the propagation of BFS access in $G(n,p)$ random graphs. Heath and Lavinus [21] proved that if the degree of a random graph is slightly greater than the logarithm of n, the number of cut edges in an optimal partition is almost the same as the number of cut edges in a random partition. In a $G(n,p)$ graph, an edge has the same probability of connecting any pair of nodes, so in this model we can ignore the edge distribution, a problem we will consider in the second model.

1) Throwing Balls into Bins: Consider the analogy in which nodes are bins, and accessing r neighbors is equivalent to throwing r balls into r bins. Given a vertex distance m, we derive the corresponding reuse distance as follows.

Starting with n empty bins, we throw r balls into r bins at each step, and repeat for m steps. The reuse distance is the number of bins that have at least one ball after the m steps.

There are $(\mathrm{C}_n^r)^m$ different cases. Let $T(m,k)$ denote the number of different cases where there are k bins with at least one ball after the mth step. We have

$$T(m,k) = \sum_{i=0}^{r} T(m-1,k-i) \cdot \mathrm{C}_{k-i}^{r-i} \cdot \mathrm{C}_{n-k+i}^{i}$$

The probability of having k nonempty bins is

$$\mathrm{P}(m,k) = \frac{T(m,k)}{(\mathrm{C}_n^e)^m}$$

$T(m,k)$ is a recursive series. In the base case when $r=1$, we have

$$T(m,k) = T(m-1,k-1)\cdot(n-k+1) + T(m-1,k)\cdot k$$

$$T(m,k) = \mathrm{C}_n^k \cdot \mathrm{S}_m^k \cdot k!$$

where $\mathrm{S}_m^k = \mathrm{S}_{m-1}^{k-1} + \mathrm{S}_{m-1}^{k} \cdot k$ is the Stirling number of the second type, which is the number of ways to partition m elements into k non-empty sets.

2) Predicting Reuse Distance: In the percolation model, m is the vertex distance, r is the average degree of the graph, and k is the reuse distance. We compute $\mathrm{P}(m,k)$ for each vertex distance m and reuse distance k and transform the vertex distance histogram to reuse distance histogram by multiplying the former with the $\mathrm{P}(m,k)$ matrix. Let $RD(d)$ is the number of accesses with reuse distance d. We compute it by

$$RD(d) = \sum_{i=1}^{n} VD(i) \cdot \mathrm{P}(i,d)$$

The solution is similar to the one used by Shen et al. [30]. Although there is no closed-form solution to $T(m,k)$, the recursive equation may be computed with a low cost as shown in the previous work.

3) Predicting BFS Level: In BFS, nodes are visited in the order of the spanning sequence. They are divided level by level in the BFS hierarchy. Let s_l be the number of nodes in level l: $s_l = |\{v : Level(v) = l, v \in V\}|$. Naturally we have the size of level 0 $s_0 = 1$ (the root node) and the total size of all levels $\sum_l s_l = n$.

For a $G(n,p)$ graph, the size of the next level s_{l+1} can be computed from the size of this and previous levels. Let $A_l = \sum_{i=0}^{l} s_i$ be the number of nodes that have been assigned a level, and $n - A_l$ the number of remaining nodes. A member of the remaining nodes will enter level $l+1$ if it has an edge to a node in level l but no edge to the earlier levels. Otherwise it would either be unreachable from level i or already have a level assigned. A node x has a probability p to have an edge to another node. The probability is $(1-p)^k$ to have no edge to a group of k nodes, and $1-(1-p)^k$ to have at least one edge to k nodes. The size s_{l+1} can be calculated by adding the probability of each of the remaining $n - A_l$ nodes being in level $l+1$:

$$s_{l+1} = \sum_{k=0}^{n-A_l} k \cdot \mathrm{C}_{n-A_l}^{k} \left(1-(1-p)^{s_l}\right)^k \cdot \left((1-p)^{s_l}\right)^{n-A_l-k}$$

Through some mathematical manipulations elided here, we derive a closed form for the summation:

$$s_l = \begin{cases} 1 & \text{if } l = 0 \\ (n - A_{l-1}) \cdot (1-(1-p)^{s_{l-1}}) & \text{otherwise} \end{cases}$$

The input for the level prediction model includes only n, p. It does not need to measure or estimate the vertex distance. As we will see in evaluation, this percolation model is accurate in predicting the reuse distance and the BFS level for random graphs but not for non-uniform R-MAT graphs.

C. The Distribution Model for R-MAT

In this section, we describe a model that considers the edge and node distribution in order to convert vertex distance into reuse distance for non-uniform graphs.

1) Modeling the Edge Distribution: The percolation model assumes uniformity, that is, a node has an equal probability connecting to any other node. A single parameter, the number of neighbors r, is used for deciding how many nodes are reached at each step. R-MAT graphs may or may not be uniform depending on the input parameter matrix. A sub-graph may contain more edges than the rest of the graph. In other words, an edge may have a higher probability connecting to one group of nodes than to another group.

To model the edge distribution, we use a graph partitioning package, METIS, to partition a graph into sub-parts. METIS uses heuristics to minimize the number of cross-partition edges (min-cut). The edge distribution manifests by the different connection probability within each part and between sibling parts.

For a first-order approximation of the edge distribution, we partition a graph into two equal-size sub-parts (by using METIS with parameter 2). Let N_1 and N_2 denote the two node sets. Let e_1 be the percentage of edges connecting two nodes in N_1, e_2 be the percentage of edges connecting two nodes in N_2, and $e_{12} = e_{21}$ be the half of the percentage of the remaining edges $(1 - e_1 - e_2)$ connecting a node in N_1 and a node in N_2. Hence, the four parameters characterize the top-level edge distribution. Figure 3 illustrates the extended model.

Figure 4 shows the edge distribution for the three R-MAT graphs. In graph *1111*, if we choose a vertex in N_1, it has probability $31.8/(31.8+18.1) = 64\%$ connecting to a node in N_1 and the remaining 36% to a node in N_2. In the other

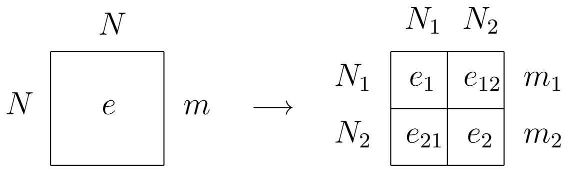


Fig. 3. Illustration of the edge distribution

31.8	18.1
18.1	32

40.6	10.2
10.2	38.9

42.3	7.6
7.6	42.3

Fig. 4. The edge distribution of the three R-MAT graphs

$$\frac{1}{N_1} \| \binom{e_1\ e_{12}}{N_1\ N_2} \| \overbrace{\binom{e_1\ e_{12}}{N_1\ N_2} \cdots \binom{e_1\ e_{12}}{N_1\ N_2}}^{e_1} \overbrace{\binom{e_{21}\ e_2}{N_1\ N_2} \cdots \binom{e_{21}\ e_2}{N_1\ N_2}}^{e_{12}} \| \cdots$$

Fig. 5. Illustration of the vertex distribution

two graphs, the probability of an intra-N_1 connection is 81% and 85% respectively for graphs *4114* and *7117*.

2) Modeling the Vertex Distribution: The second problem not considered in the percolation model is that the vertices within a vertex distance may not belong to each sub-graph with an equal probability. This is a result of the edge distribution.

For example, suppose that a BFS tree is rooted in N_1. The BFS spanning sequence is depicted in Figure 5. The sequence is divided by $\|$ between different BFS levels. In the first level (BFS level=0), there is only one vertex in N_1. In the second level (BFS level=1), there are more nodes from N_1 than from N_2 since $e_1 > e_{12}$. In the third level, all four parameters of the edge distribution, e_1, e_2, e_{12}, e_{21}, affect the composition of nodes.

To consider the vertex distribution, we extend the percolation model. For each vertex distance, we conduct the m steps in two phases. The first has m_1 steps, and the second m_2 steps ($m = m_1 + m_2$). We use the edge distribution to derive the vertex distribution. From Figure 5, we use the proportion $m_1/m_2 = e_1/e_{12}$ when computing the distribution in N_1 nodes and $m_1/m_2 = e_{21}/e_2$ among the N_2 nodes.

In each step in the m_1 phase, e_1 bins are chosen from N_1, and e_{12} bins are chosen from N_2. Then $e = e_1 + e_{12}$ balls are put into these bins. Similarly, in each step of the second phase, e_{21} bins are chosen from N_1 and e_2 bins are chosen from N_2, and these bins are filled by $e = e_2 + e_{21}$ balls.

3) Put It Together: The distribution model answers the question how many bins that have at least one ball after m steps. There are $(\mathrm{C}^{e_1}_{n/2})^{m_1} \cdot (\mathrm{C}^{e_{12}}_{n/2})^{m_2}$ different cases for the N_1 set and $(\mathrm{C}^{e_2}_{n/2})^{m_2} \cdot (\mathrm{C}^{e_{21}}_{n/2})^{m_1}$ different cases for the N_2 set. Let $T_1(j,k)$ denotes the different cases that N_1 has k bins with at least one ball after j steps, we have a revised recurrence equation.

$$T_1(j,k) = \begin{cases} \sum_{i=0}^{e_1} T_1(j-1,k-i) \cdot \mathrm{C}^{e_1-i}_{k-i} \cdot \mathrm{C}^{i}_{\frac{n}{2}-k+i} & j \le m_1 \\ \sum_{i=0}^{e_{12}} T_1(j-1,k-i) \cdot \mathrm{C}^{e_{12}-i}_{k-i} \cdot \mathrm{C}^{i}_{\frac{n}{2}-k+i} & j > m_1 \end{cases}$$

Let $\mathrm{P}_1(m,i)$ denotes the probability that N_1 has i non-empty bins after m steps. Then we have

$$\mathrm{P}_1(m,i) = \frac{T_1(m,i)}{(\mathrm{C}^{e_1}_{n/2})^{m_1} \cdot (\mathrm{C}^{e_{12}}_{n/2})^{m_2}}$$

$\mathrm{P}_2(m,i)$ can be computed similarly. The probability that k bins have at least one ball after m steps is

$$\mathrm{P}(m,k) = \sum_{i=0}^{k} \mathrm{P}_1(m,i) \cdot \mathrm{P}_2(m,k-i)$$

4) Prediction of BFS Levels and Reuse Distance: The distribution model can predict the size of the BFS tree hierarchy, namely, the number of BFS levels and the number of vertices at each level. We can predict the number ($Level[LevelCnt]$) of vertices in each BFS level from Algorithm 1. It is obvious that $Level[0] = 1$ and $Level[1] = Degree[root]$. The experiment results show that the BFS levels are similar when the degrees of roots are close. Next we predict BFS hierarchy parameterized by the degree of the chosen root.

If the nodes of the previous level access t nodes and there are a nodes that has already been accessed, there will be $t(n-a)/n$ nodes in the next level. Let A_l denotes the number of accessed nodes before the $(l+1)$th level. We choose $Level[l]$ such that the probability of $\mathrm{P}(Level[l-1], Level[l] \cdot n/(n - A[l]))$ is the largest.

D. The Vertex Distance in Random Graphs

In the third model, we show the relation between the vertex distance and the high-level graph parameters. A random graph has two parameters: size n and connection probability p. The following theoretical analysis shows that the vertex distance in BFS follows the geometric distribution, that is, the portion of vertex distances of length k is $(1-p)^{k-1}p$.

Lemma 1: Let G be a random graph from the Erdős-Rényi random graph model $\mathcal{G}_{n,p}$. Assume $p = \omega(1/n)$ as $n \to \infty$. With probability $1 - o(1)$ the distribution of the vertex distances of the *Level* array in the BFS algorithm is, up to $o(1)$ ℓ_1-error, geometric distribution with parameter p.
Proof : For $v \in V$ let $T_v = i$ if the i-th `Dequeue` operation returns v (thus, for example, for the root r we have $T_r = 1$). Let X_k be the number of vertices that are marked visited (that is, have Level[v] $< \infty$) just before k-th `Dequeue` operation is executed. Thus

$$X_1 = 1.$$

Let Z_k be the number of vertices that are discovered from the vertex v that is returned by the k-th *Dequeue* operation (that is, the vertex with $T_v = k$). There are $n - X_k$ vertices that can be potentially discovered (that is, have Level$[w] = \infty$) and each is discovered with probability p. Thus

$$Z_k \sim \text{Binomial}(n - X_k, p).$$

We then have

$$X_{k+1} = X_k + Z_k.$$

The process ends when $X_{F+1} = F$, that is, all discovered vertices were already popped from the queue (here F is the number of vertices discovered by the BFS algorithm—it is not necessarily equal to n, since the random graph can be disconnected[1]).

Let $v \in V$ be a vertex (discovered by the BFS algorithm). Assume that v is not the root (we can ignore the case when v is the root r, since with probability $1 - o(1)$ the accesses to `Level[r]` form an $o(1)$ fraction of the accesses to the `Level` array). Let w be the vertex from which we discovered v. Note that

- there is no edge between u and v if $T_u < T_w$ (otherwise v would be discovered from u),
- each vertex with T_u value in $\{T_w + 1, \ldots, X_{T_v}\} \setminus \{T_v\}$ is a neighbor of v, with probability p (these events are independent), and
- there are Z_{T_v} additional neighbors of v (these are the vertices discovered from v).

We are going to ignore all the accesses arising from the vertices discovered from v (since aggregated over all v there are only $O(n)$ such accesses—a drop in the bucket compared to the $\omega(n)$ other accesses (more precisely, we have $\omega(n)$ of them with probability $1 - o(1)$)).

Now we are going to add $O(n)$ "made-up accesses" to make the distribution of the vertex distances equal to the geometric distribution (note that this is again just a drop in the bucket). After the sequence of potential vertices with T_u value in $T_w + 1, \ldots, X_{T_v}$ add more "virtual vertices" until we obtain one that is connected to v. Now the vertex distances are exactly geometrically distributed, and since we obtained this by $o(1)$ change in the distribution it must be that the true distribution of vertex distances is distance $o(1)$ from the geometric distribution. ■

Figure 6 shows the distribution of vertex distances for a random-graph BFS traversal. When n is large (1000 in this case), the prediction, given by the geometric distribution, closely matches with the measurement.

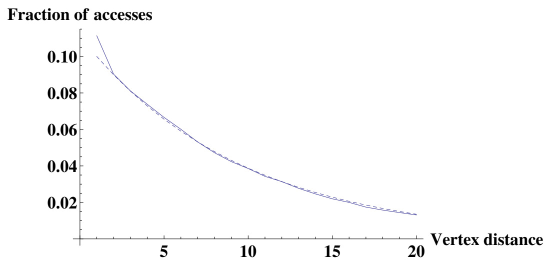


Fig. 6. The vertex distance distribution for a random $G \sim \mathcal{G}_{n,p}$ compared to the geometric distribution ($n = 1000, p = 0.1$).

IV. Evaluation

In evaluation, we first measure the reuse distance histogram (MRD). We then apply the first two models to use the vertex distance histogram to obtain the predicted reuse distance histogram (PRD). We use the formula proposed by Shen et al. [30] to calculate the prediction accuracy as follows:

$$accuracy = 1 - \frac{\sum_i |MRD_i - PRD_i|}{2}$$

where MRD_i is the percentage of references in the ith bin in MRD and PRD_i is the percentage in the ith bin of PRD.

Through experimental results we found that the reuse distance histograms and BFS levels are similar for BFS starting from different roots in the input graph. The difference is very small so we do not show it in the figures (as we did in Figure 2).

TABLE II
$G(n,p)$ Matrices,$n = 16384, \ln 16384 \approx 10$

p	ef	m	a	b	c	d
0.001	32	268156	0.19	0.25	0.25	0.31
0.002	65	537076	0.21	0.25	0.25	0.29
0.005	164	1342400	0.22	0.25	0.25	0.28
0.010	328	2685853	0.23	0.25	0.25	0.27
0.015	492	4027896	0.23	0.25	0.25	0.27
0.020	656	5370210	0.24	0.25	0.25	0.26

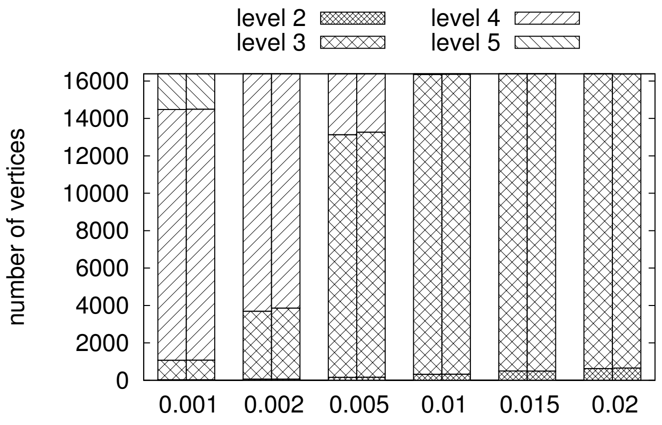


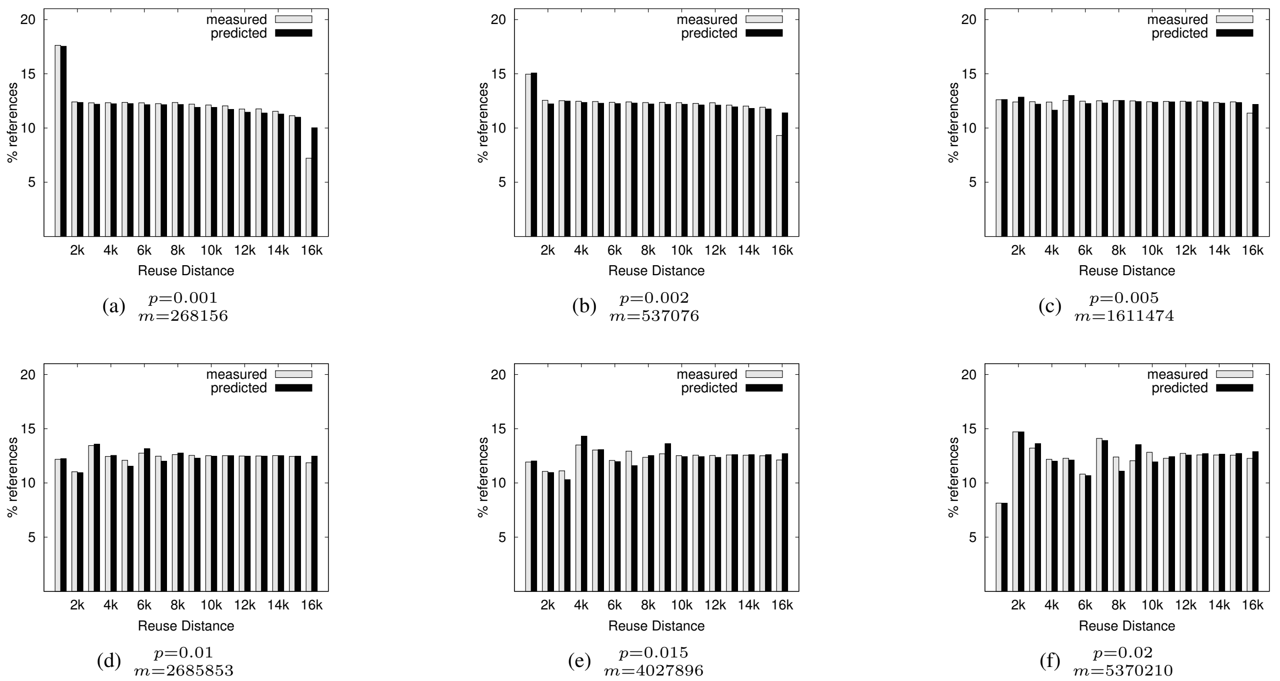
Fig. 7. Measured and Predicted BFS Levels for *G(16384,p)*

A. Random Graph Results

We use the *GTgraph* designed by Madduri and Bader [32] to generate random graphs with varying p parameters. Table II lists six graphs with number of nodes $n = 16384$. We use METIS to partition, each of these graphs into two parts. Then the adjacency matrix is split into four parts each of which has a percentage of all edges. The four parameters a, b, c and d like the ones used in the extended model for RMAT graphs. From Table II we can see that the four parts become equal as the parameter p increases, showing the property proved by Heath and Lavinus [21]. Since the edge distribution is uniform, we can expect the percolation model to predict well (even though it does not consider the edge distribution).

The six graphs in Figure 8 show the measured and predicted reuse distance results for graphs generated by the $G(n,p)$

[1]In fact, the threshold for connectedness is $p = (\log n)/n$, see, e. g., [31], p. 164.



(a) p=0.001 m=268156
(b) p=0.002 m=537076
(c) p=0.005 m=1611474
(d) p=0.01 m=2685853
(e) p=0.015 m=4027896
(f) p=0.02 m=5370210

Fig. 8. Measured and Predicted BFS Reuse Distance for *G(16384,p)*

model, with $n = 16384$ and six values of p ranging from 0.001 to 0.02. The prediction accuracy is over 97% for every graph. The average accuracy of the six graphs is 98%.

We consider an extreme case. If the graph is a complete graph, namely $p = 1$ and $m = (n^2 - n)/2$ (we ignore the self edges), all the m accesses will have reuse distance of n. There will be $m - n$ accesses with vertex distance 1 and n accesses with vertex distance 2. We can see the trend from Figure 8 that the shape of the reuse distance histogram approaches uniform distribution, which is that of the extreme case.

Figure 7 shows the the measured and predicted BFS levels results for graphs generated by $G(n,p)$ model. The accuracy is even higher, with the average of the six graphs at 99%.

B. R-MAT Graph Results

Random graphs are uniform in that there are no unevenly connected sub-graphs (i.e. small-world communities) [21]. The percolation model, although accurate for random graphs, does not provide an accurate prediction for R-MAT graphs. For these non-uniform graphs, we need the distribution model.

The three histograms in Figure 9 show the measured and predicted reuse distance of the three R-MAT graphs. As we analyzed, the reuse distance predicted from vertex distance is larger than the measured distance especially when the vertex distance increases. The three graphs are predicted with an accuracy of 96%, 95%, and 94%.

Figure 10 shows the measured and predicted BFS hierarchy in the three graphs. We sample 10% of vertices in each graph and record the BFS level for each vertex. After analyzing the results we found that BFS levels are similar when the degrees of roots are closer. So we plot the average number of vertices for levels larger than 1 and use the degree of the root as the x axis. The average prediction accuracy of the three graphs is 96%.

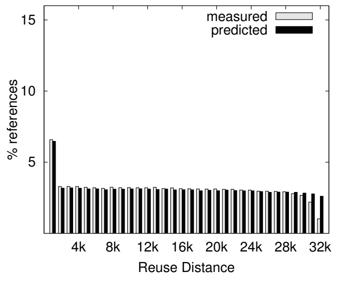
C. The Effect of Graph Topology

The three R-MAT graphs have the same size but different topologies in terms of how nodes are connected. In *1111*, edges are evenly distributed. In *4114* and *7117*, edges are more concentrated in some sub-graphs than in others.

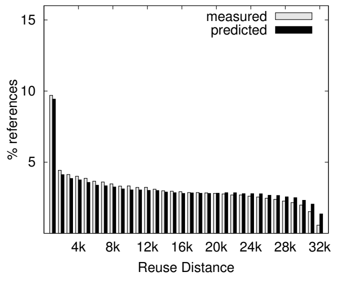
The topology matters, as shown by Figure 9 for the reuse distance distribution and by Figure 10 for the BFS hierarchy. The uniform edge distribution has the least locality. The reuse distance has a nearly uniform distribution. Interestingly and perhaps not surprisingly, this result is similar to the reuse distance distribution of random data access [33]. As edges are more concentrated in sub-graphs, the locality improves.

The effect of edge concentration also changes the shape of the BFS hierarchy, in particular, the number of levels and the size of each level. A uniform edge distribution yields the shallowest hierarchy. As edges are more concentrated, the number of levels increases. Generally speaking, randomness means flatness, and locality means hierarchy.

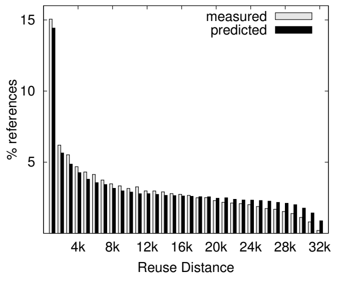
It is clear from the presented results that the edge and the node distributions affect locality and BFS hierarchy. The predictions for the *1111* graph would be quite inaccurate for the *4114* and especially the *7117* graph. The benefit and the effect of the distribution model are also clear from these figures.



(a) 1111

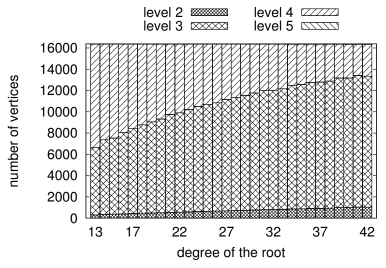


(b) 4114

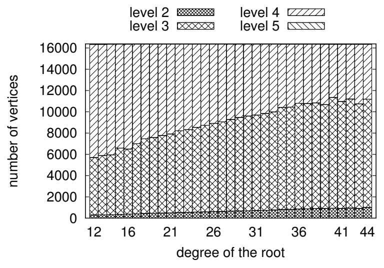


(c) 7117

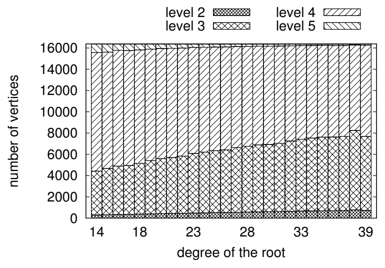
Fig. 9. Measured and Predicted BFS Reuse Distance for *R-MAT*



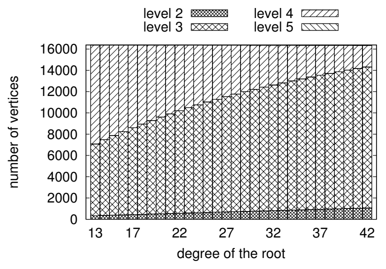
(a) Measured Results of 1111



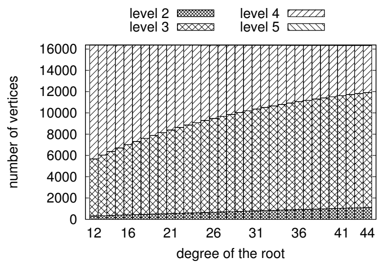
(b) Measured Results of 4114



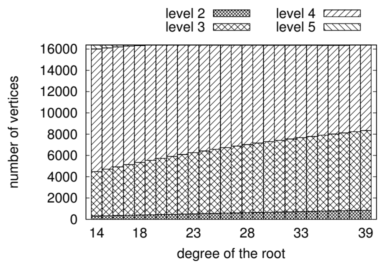
(c) Measured Results of 7117



(d) Predicted Results of 1111



(e) Predicted Results of 4114



(f) Predicted Results of 7117

Fig. 10. Measured and Predicted BFS Levels of R-MAT graphs. The level partition is sensitive to the root degree.

V. Summary and Future Work

In this paper we have defined a concept called the vertex distance and described three models. The percolation model shows how time in vertex distance maps to locality in reuse distance in uniformly random graphs. In non-uniform graphs, the edge distribution has a strong impact to locality and the BFS hierarchy in addition to the effect of graph size. The distribution model approximates the imbalance in an equal partition of a graph. Finally, we have proved that the vertex distance in random graphs follows a geometric distribution. The evaluation results with random and R-MAT graphs show accurate prediction by the three models.

For random graphs, the new models can predict the locality given just two parameters, n and p. For R-MAT graphs, the distribution model requires a profiling run to measure the vertex distance and a METIS run to partition the input graph into two parts. While the overhead is not much less than measuring reuse distance directly, the analysis provides an explanation which the observation itself does not provide. We are extending the third model to predict the vertex distance without profiling. Currently the edge distribution is obtained by dividing a graph once. It works well for the R-MAT graphs used in this paper but may not be precise enough in general. The formula of the distribution model can be extended to include finer partitions, but the cost of its evaluation increases rapidly. We are studying simpler formulations and approximate solutions. In addition, we are considering techniques to remove the need for running METIS, for example by sampling instead of running the analysis on the full graph.

Finally, the concept of vertex distance is based on a spanning sequence in BFS. The same type of sequences exist in other graph traversal algorithms. We are studying vertex-distance based models in other algorithms. The result may allow us to compare and explain the difference in locality for example between breadth-first and depth-first graph traversals.

In science and engineering computing, two common classes of problems are n-body simulation and sparse-matrix solvers. Prior work has shown significant benefits from run-time computation and data reorganization [34]–[39]. One direction of this work is to study efficient locality probes to help a run-time system choose when and how to apply these transformations.

Acknowledgments

The authors would like to thank Xiaoya Xiang, Jacob Brock, Bin Bao, Tongxin Bai and the anonymous reviewers for their helpful comments. This work is partially supported by National S&T Major Projects: Core Electronic Devices, High-end General Chips and Fundamental Software (No.2009ZX01036-001-002), National Natural Science Foundation of China (No.61133005), the National 863 Plan of China (No. 2009AA01A129, No. 2009AA01A134), the National High-tech R&D Program of China (No. 2012AA010903, No. 2012AA010902), the US National Science Foundation (Contract No. CCF-1116104, CCF-0963759, CCF-0910415), the IBM Center for Advanced Studies, and Huawei Corporation.

References

[1] D. A. Bader and K. Madduri, "Designing multithreaded algorithms for breadth-first search and st-connectivity on the cray mta-2," in *ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing*. IEEE Computer Society, 2006, pp. 523–530.

[2] Y. Zhang and E. A. Hansen, "Parallel breadth-first heuristic search on a shared-memory architecture," in *AAAI-06 Workshop on Heuristic Search, Memory-Based Heuristics and Their Applications*, 2006.

[3] Y. Xia and V. K. Prasanna, "Topologically adaptive parallel breadth-first search on multicore processors," in *PDCS '09: the 21st International Conference on Parallel and Distributed Computing and Systems*, 2009.

[4] K. You, J. Chong, Y. Yi, E. Gonina, C. Hughes, Y.-K. Chen, W. Sung, and K. Keutzer, "Parallel scalability in speech recognition," *Signal Processing Magazine, IEEE*, vol. 26, no. 6, pp. 124 –135, 2009.

[5] V. Agarwal, F. Petrini, D. Pasetto, and D. A. Bader, "Scalable graph exploration on multicore processors," in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010, pp. 1–11.

[6] C. E. Leiserson and T. B. Schardl, "A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers)," in *SPAA '10: Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*. ACM, 2010, pp. 303–314.

[7] R. Pearce, M. Gokhale, and N. M. Amato, "Multithreaded asynchronous graph traversal for in-memory and semi-external memory," in *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010.

[8] V. Muntés-Mulero, N. Martínez-Bazán, J.-L. Larriba-Pey, E. Pacitti, and P. Valduriez, "Graph partitioning strategies for efficient bfs in shared-nothing parallel systems," in *WAIM '10: Proceedings of the 2010 international conference on Web-age information management*. Springer-Verlag, 2010, pp. 13–24.

[9] A. Buluc and K. Madduri, "Parallel breadth-first search on distributed memory systems," *arXiv '11*.

[10] S. Idwan and W. Etaiwi, "Computing breadth first search in large graph using hmetis partitioning," *European J. of Scientific Research*, pp. 215–221, 2009.

[11] A. Yoo, E. Chow, K. Henderson, W. McLendon, B. Hendrickson, and U. Catalyurek, "A scalable distributed parallel breadth-first search algorithm on bluegene/l," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2005, p. 25.

[12] L. Luo, M. Wong, and W.-m. Hwu, "An effective gpu implementation of breadth-first search," in *DAC '10: Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 52–55.

[13] Y. S. Deng, B. D. Wang, and S. Mu, "Taming irregular eda applications on gpus," in *ICCAD '09: Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 539–546.

[14] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the gpu using cuda," in *HiPC'07: Proceedings of the 14th conference on High performance computing*. Springer-Verlag, 2007, pp. 197–208.

[15] K. Mehlhorn and U. Meyer, "External-Memory Breadth-First Search with Sublinear I/O," in *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*. Springer-Verlag, 2002, pp. 723–735.

[16] K. Munagala and A. Ranade, "I/O-complexity of graph algorithms," in *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1999, pp. 687–694.

[17] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Syst. J.*, vol. 9, pp. 78–117, June 1970.

[18] M. D. Hill and A. J. Smith, "Evaluating associativity in cpu caches," *IEEE Trans. Comput.*, vol. 38, no. 12, pp. 1612–1630, 1989.

[19] Y. Zhong, X. Shen, and C. Ding, "Program locality analysis using reuse distance," *ACM Trans. Program. Lang. Syst.*, vol. 31, no. 6, pp. 1–39, 2009.

[20] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *SDM '04: SIAM Data Mining*, 2004.

[21] J. L. Ganley and L. S. Heath, "Optimal and random partitions of random graphs," *The Computer Journal*, vol. 37, pp. 641–643, 1994.

[22] C. Cascaval and D. A. Padua, "Estimating cache misses and locality using stack distances," in *ICS '03: Proceedings of the 17th annual international conference on Supercomputing*, 2003, pp. 150–159.

[23] A. Dan and D. Towsley, "An approximate analysis of the lru and fifo buffer replacement schemes," in *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, In SIGMETRICS '90. ACM, 1990, pp. 143–152.

[24] E. N. Gilbert, "Random graphs," *Annals of Mathematical Statistics*, vol. 30, pp. 1141–1144, 1959.

[25] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[26] M. A. Hassaan, M. Burtscher, and K. Pingali, "Ordered vs. unordered: a comparison of parallelism and work-efficiency in irregular algorithms," in *PPoPP '11: Proceedings of the 16th ACM symposium on Principles and practice of parallel programming*. ACM, 2011, pp. 3–12.

[27] A. Buluc and J. Gilbert, "On the representation and multiplication of hypersparse matrices," in *IPDPS '08: IEEE International Symposium on Parallel and Distributed Processing*, april 2008, pp. 1–11.

[28] D. Ajwani, R. Dementiev, and U. Meyer, "A computational study of external-memory bfs algorithms," in *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. ACM, 2006, pp. 601–610.

[29] G. Cong and S. Sbaraglia, "A study on the locality behavior of minimum spanning tree algorithms," in *HiPC '06: International Conference on High Performance Computing*, 2006, pp. 583–594.

[30] X. Shen, J. Shaw, B. Meeker, and C. Ding, "Locality approximation using time," in *POPL '07: Proceedings of the ACM symposium on Principles of programming languages*. ACM, 2007, pp. 55–61.

[31] B. Bollobás, *Random graphs*, In Cambridge studies in advanced mathematics. Cambridge University Press, 2001.

[32] K. Madduri and D. A. Bader, "Gtgraph: A suite of synthetic random graph generators," *http://sdm.lbl.gov/ kamesh/software/GTgraph/*.

[33] X. Gu and C. Ding, "Reuse distance distribution in random access," *TR930, Computer Science Dept., U. Rochester*, 2008.

[34] M. M. Strout, L. Carter, and J. Ferrante, "Compile-time composition of run-time data and iteration reorderings," in *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*. ACM, 2003, pp. 91–102.

[35] C. Ding and K. Kennedy, "Improving cache performance in dynamic applications through data and computation reorganization at run time," in *PLDI '99: Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation*, 1999, pp. 229–241.

[36] N. Mitchell, L. Carter, and J. Ferrante, "Localizing non-affine array references," in *PACT '99: Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, Oct. 1999.

[37] H. Han and C. W. Tseng, "Improving locality for adaptive irregular scientific codes," in *LCPC'00*, Aug. 2000.

[38] H. Han and C. Tseng, "Exploiting locality for irregular scientific codes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 7, pp. 606–618, 2006.

[39] J. Mellor-Crummey, D. Whalley, and K. Kennedy, "Improving memory hierarchy performance for irregular applications," *International Journal of Parallel Programming*, vol. 29, no. 3, Jun. 2001.