F.A.R Framework

F.A.R Framework

A binary framework where information retrieval is **F**ast **A**ccurate and **R**elevant (**F.A.R**)

Gautham B A
Dept: Computer Science
College-PES Institute of Technology, BSK 3rd stage
Bangalore, India
e-mail:gautham.bangalore@gmail.com

Aditya Agarwal
Dept: Computer Science
College-PES Institute of Technology, BSK 3rd stage
Bangalore, India
e-mail:skymanaditya1@gmail.com

Abstract— **Data Resource Management is the development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle needs of an enterprise [1]. Retrieving relevant and meaningful information as specified in the search query from the ocean of data, along with the retrieval being efficient with regard to memory and computation is the purpose of the proposed method. If related data is organized according to the proposed method, all the set operations such as union, intersection, difference and cross-product can be performed in *O(n)* without having the overhead of maintaining a sorted list or any other constraints. Every Boolean Algebra expression can be translated to a set-relation equation. Since the proposed framework is compatible with any set-relation operation, with the help of translation, Boolean Algebra operations can be performed. Every Boolean algebra operation has a meaning in the proposed framework and hence, innumerable meaningful information can be obtained with every different combination of Boolean expression.**

Keywords—Data management; boolean algebra, set, set-relation, adjacency matrix, upper-triangular matrix, binary and-or-exclusive or operations, time complexity, Big-Oh, space complexity.

I. INTRODUCTION

Data is a set of values of qualitative or quantitative variables; restated, pieces of data are individual pieces of information [2]. The order of measure of data has been changing from the very genesis of computers. It was usually around a few bytes initially and gradually the measure grew to a few megabytes for the whole company and as time progressed, we have reached a stage where each individual possesses a terabyte of data on an average. Raw data (also known as primary data) is a term for data collected from a source. Raw data has not been subjected to processing or any other manipulation [4]. Unfortunately, most of the data that the people store are raw data. Putting a constraint over the amount of raw data that each person can store on a website would mildly offend their reputation. So, an alternative is to come up with a method to handle this. [3] tries to convince the need for an innovation that would aid the accessing huge data swiftly, and also, the acquisition being efficient and easy at the same time.

The proposed method provides a framework for arranging the data in the form of an adjacency matrix as shown in Fig(1) and also to express the relationship between the elements. Since the adjacency matrix is symmetric, the elements that lie beyond one half of the principal diagonal elements can be ignored for making it more efficient with regard to memory as described in Section V. So, the resulting framework will only consist of an upper-triangular matrix as shown in Fig(6).

The implementation of the proposed method is explained by in the context of a social networking site. Here, the relationship defined between two elements *A* and *B* is "*friend*".

With the help of the proposed method, finding common friends between persons *A* and *B*, translates to performing intersection between sets *A* and *B* in the proposed framework and can be done in *O(n).* Similarly, finding all friends of persons *A* and *B*, which, in the proposed framework translates to finding the union of sets *A* and *B*, can be done in *O(n).* It should be noted that there is no effort needed to maintain a sorted list and yet the above two operations are being done in *O(n)* as explained in Section IV.D.

In the same way as stated above, any Boolean expression can be translated to a set-relation expression and can be applied to the proposed framework to obtain a meaningful and more relevant search result. For example, a Boolean expression *A.B.C* (which is read as *A conjunction B conjunction C*) is translated to the set-relation operation as *A intersection B intersection C.* Thus, every Boolean operation performed on the framework will have a meaning which is obtained in the resulting bit-string.

The paper is organized as follows - 1. Introduction 2. Literature survey 3. Ease of use 4. Design and implementation 5. Improvements 6. Results 7. Conclusion 8. References.

II. LITERATURE SURVEY

1. *Social Referral: Leveraging network connections to deliver recommendations*[8]

Much work has been done to study the interplay between recommender systems and social networks. This creates a very powerful coupling in presenting highly relevant recommendations to the users. What this work achieves is-
1) A novel recommendation delivery paradigm called Social Referral, which utilizes a user's social network for the delivery of relevant content. 2) An implementation of the paradigm is described in a real industrial production setting of a large online

professional network. 3) A study of the interaction between the trifecta of the recommender system, the trusted connections and the end consumer of the recommendation by comparing and contrasting the pro- posed approach’s performance with the direct recommender system.

The proposed framework handles the recommendation of friends in a very different and efficient way as described in Section IV.E. It does not rely upon any the data from an external source in order to determine the relationship between the entities in the framework.

2. *User Interests Identification on Twitter Using a Hierarchical Knowledge Base*[9]

Analysis of user generated content for personalization and recommendation tasks and identifying user interests from tweets. Semantic enrichment of Twitter posts, to determine user interests. These approaches typically use available public knowledge bases (such as Wikipedia) to spot entities and create entity based user profiles. Hierarchical relationships present in knowledge-bases to infer user interests expressed as a Hierarchical Interest Graph. Focus on using current browsing session for providing personalization and therefore a lack of identification of the broader interests. Primitive Interests – the entities that can be directly spotted from user's tweets are called as Primitive Interests. Hierarchical Interests – the entities that exploit the knowledge linked to Primitive Interests in Wiki to determine the user's interest in broader categories.

Twitter page and followers recommendations – It uses information it collects about users' browsing habits across all sites with Twitter share buttons to recommend accounts to follow. Every time you visit a site that has a follow button, a tweet this button or a hover-card, Twitter records your behavior and recommends the pages and people to follow in the next 10 days.

In the proposed framework, we use binary operations *and, or, exclusive-or* to determine the mutual friends, friend recommendations and recommended page likes. Since the framework is represented using adjacency matrix and user relationship are depicted using 0s and 1s, in order to find the mutual friends, the ‘*binary and’* operation of the two lists is carried out to determine the result as described in Section IV.E .

3. *Google's Pregel PageRank*[10]

Page Rank : A ranking is based on PageRank, a graph algorithm computing the importance of webpages. PageRank p is the principal eigenvector of the Markov matrix M defined by the transition probabilities between web pages. It can be obtained by iteratively multiplying an initial PageRank vector by M(Power Method) : Pi+1 = Mpi which has the following drawbacks-

1. *Non intuitive:* only crazy scientists think in matrices and Eigen vectors.
2. *Unnecessarily slow*: Each iteration is a single MapReduce job with lots of overhead.
3. *Hard to implement*: a join has to be implemented by hand, lots of work, best strategy is data dependent.

On the other hand, in the proposed framework we are using simple and traditional adjacency matrix as shown in Fig(1) to represent the framework consisting of users and page likes as shown in Fig(4). While we are performing *binary and, or* and *exclusive-or* operations to find the rank of various pages we are able to achieve several things such as those described below-

1 *Flexibility*: As opposed to a single method for computing the page rank, our framework utilizes the power of binary representation and computes the page rank on various parameters like the maximum likes on a page, maximum friends likes on a page and maximum likes by common friends.
2. *Essentially easy*: Since the efficiency of performing these operations is of the order of n as described in Section IV.D, the proposed framework is tremendously fast.
3. *It is simple*: Since we represent data using only adjacency matrix and use binary operations to perform the computation, the proposed framework is relatively simpler as the operations are performed on the bit-levels.

III. Ease of Use

Though the proposed method is described in the context of a social networking site, its application spans over a large number of areas. It can be applied at all places where related data is stored, such as, all places where a relational database is used and at all places where set operations have to be performed, to recommend the most relevant sites to people, suggesting friends. Also, the proposed framework guarantees that any set operation performed between any two sets will not exceed *O(n)* as described in Section IV.D. The framework makes use of a simple adjacency matrix for its representation as shown in Fig(1) and the operations as described in Section IV.B are extremely simple but powerful. The information that is retrieved is fast, accurate and relevant as the title states.

IV. Design and Implementation

The description of the design is done in the context of a social networking site only for the purpose of easy understanding.

Fig(1) shown below displays an adjacency matrix of people numbered 1 to 7 and their relationship with every other person.

	1	**2**	**3**	**4**	**5**	**6**	**7**
1st bit	0	1	1	0	1	1	0
2nd bit	1	0	0	1	1	0	1
3rd bit	1	0	0	1	0	1	1
4th bit	0	1	1	0	1	1	0
5th bit	1	1	0	1	0	0	1
6th bit	1	0	1	1	0	0	0
7th bit	0	1	1	0	1	0	0

Fig(1)

A. <u>Representation of the framework</u>

Illustration:

1. The bit string under person 1 in Fig(1) is 0110110.
2. In this bit string, bits 2, 3, 5 and 6 are set.
3. This implies that persons 2, 3, 5 and 6 are friends of person 1. The other bits 4 and 7 are not set. This implies that 4 and 7 are not friends of person 1. We observe from the matrix that bit 1 is also not set. Here, the reflexive property doesn't make sense, it would be rather obscene to state that person 1 is not a friend of person 1.
4. *Thus, in the framework, all the principal diagonal bits are set to 0 as the reflexive property would not be so meaningful.*
5. Similarly, the bit string under person 5 is 1101001. This implies that persons 1, 2, 4 and 7 are friends of person 5 but 3 and 6 are not.
6. *In general, the i^{th} bit under any person j decides the relationship between person i and j. If this bit is set, it means that a relation exists between person i and j.*

B. <u>Performing operations on the framework</u>

As stated in the introduction, every Boolean expression has a meaning in the framework. Evaluation of a Boolean expression results in a bit-string again. As an example, Fig(2) shows the *binary and, or* and *exclusive-or* operations performed on the bit-strings of person 1 and 2.

	1	**2**	**1and2**	**1or2**	**1eor2**
1st bit	0	1	0	1	1
2nd bit	1	0	0	1	1
3rd bit	1	0	0	1	1
4th bit	0	1	0	0	1
5th bit	1	1	1	1	0
6th bit	1	0	0	1	1
7th bit	0	1	0	1	1

Fig(2)

1. ***Determining commonalities between two entities in the framework.***
   In terms of a social networking site, this operation maps to finding the list of common friends between two people. This translates to performing the *binary and* operation (or *set intersection)* between the bit-strings of two people. As shown in column 4 of Fig(2), only bit 5 is set. This implies that only person 5 is a friend of both the persons 1 and 2.

2. ***Determining the closure of two entities.***
   This can be done by performing the *binary or* operation(or *set-union)* between the bit-strings of two people. The result of this operation for persons 1 and 2 is shown in column 5 of Fig(3). The resulting bit-string has all the bits set except the 4th bit. The meaningful information that we can get from this is that all the people except person 4 in the framework are either friends with person 1 or person 2. Perhaps the ***complement*** of the *binary or* operation would yield a more useful information - the bits that are set in bit-string resulting from the *complement of binary or* operation would be those persons who are not friends with either of the two persons.

3. ***Determining inconsistency between two entities in the framework***
   The result of an *exclusive-or* operation is 1 only if the two operands are dissimilar and is 0 otherwise. Since the very nature of the *exclusive-or* operation determines the dissimilarity, this operation can be employed to find the inconsistency between two people. The result of this operation for persons 1 and 2 is shown in column 6 of Fig(2). Here, if there are more 1s in the resultant bit-string, it implies that there is a lot of inconsistency between persons 1 and 2. The ***complement*** of this operation would suggest the *consistency* between two people.

4. ***Handling relational transactions between two entities in the framework.***
   Two types of *relational transactions* are possible, they are *befriending* and *unfriending.* Suppose person 1 wants to befriend person 3. In this case, the bit-strings stored underneath person 1 and 3 have to be modified. The first bit of person 3 and the third bit of person 1 have to be set to one, to indicate that person 1 and 3 are now friends in the framework. Unfriending is the reverse of this operation and the corresponding bits are cleared from the bit-strings if two people unfriend each other.

5. ***Checking the existence of a relation between two entities in the framework.***
   From the perspective of a social networking site, this operation corresponds to testing whether a bit in the bit-string of a particular entity is set or not. For example, to check whether person 1 is a friend of person 2, the 2nd bit of person 1's bit-string is checked. As shown in Fig(1), 2nd bit of person 1's bit-string is a 1. Hence, it implies that person 1 and 2 are friends. This operation can also be carried out on person 2. In that case, we check whether the 1st bit under person 2's bit-string is set or not. The result will be same in both the cases as the framework is represented using an adjacency matrix which is symmetric.

Likewise, the number of meaningful and relevant results that can be obtained by the action of Boolean expression on the proposed framework is innumerable. The above illustrations are only a few attempts to explain the functions of the proposed framework. The operations can be performed between any number of people. e.g. *1.2.3.4* would result in a bit-string that contains the common friends of persons 1, 2, 3 and 4.

C. Organization in memory and representation

The framework is designed taking memory into consideration as well. Since the proposed framework stores the relations between sets in the form of bit-strings, and the memory of a computer also stores data in binary format, we can employ this similarity to make the proposed framework efficient with regard to storage.

A 32-bit machine can store a bit-string that is 32 bits long and occupies 4 bytes in one memory cell. For example, the number 54 is stored in the computer memory as a bit-string 0011 0110 with 24 zeros before. The bit-string under person 1 is 0110110 and its decimal equivalent is 54. Since the proposed framework contains information in only binary, it is sufficient if we just store the decimal equivalent of the bit-string as shown in Fig(3) instead of wasting memory by storing each bit in a single memory location, resulting in an array. The computer automatically converts the decimal number to its binary equivalent when the operations have to be performed.

1	2	3	4	5	6	7
54	89	105	54	75	13	22

Fig(3)

Thus, in a 32-bit machine, information about the relation between one person and 31 other persons can be stored in one memory location and occupies only 4 bytes.

D. Time complexity of performing the operations on framework

The proposed framework is designed by giving more prominence to time efficiency along with memory. The time efficiency of a few general operations that can be performed on the framework are stated below. The same idea can be extended to other operations as well.

1. ***Determination of closure of two entities***

   This operation requires a *binary and* operation to be performed with the bit-strings of two people as its operands. Since a 32-bit machine can perform operations on 32 bits at once, This operation can be performed in constant time(*O(1)*) if the number of entities(people) in the framework are less than 32. In the worst case, this operation can be done in *O(n)* as the framework will occupy more than one memory cell if the number of people in the framework are greater than 32. This analysis remains the same for the finding the union and also for *binary exclusive-or* operations.

2. ***Handling relational transactions***

   The two operations that are involved in this transaction can be performed in constant time (*O(1)*) irrespective of the number of people in the framework as it involves only setting or clearing a particular pair of bits. Suppose person *i* wishes to befriend person *j*. As described in Section IV.B.4, we must perform a *binary or* operation between the bit-string of person *j* and 2^i , and also on the bit-string of person *i* and 2^j. For example, if person 3 befriends person 5, we have to perform the operation *1101001 or|* 2^5. The result of this operation is *1111001*, the 5th bit of person 3's bit-string is set and also by doing *1001011 or* 2^3, 3rd bit of person 5's bit-string is set. Hence, it can be implied that person 3 and 5 are now friends. *This operation is carried out in constant time O(1) since it only performs a pair of 'or' operations.*

3. ***Checking the existence of a relation between the entities***

   Suppose we wish to know whether person *i* is a friend of person *j*. As described in Section IV.B.5, we must perform a *binary and* operation between the bit-string of person *j* and 2^i. For example, to check whether persons 3 and 5 are friends, we have to perform the operation *1101001 &* 2^3. The result of this operation is 0 and hence, it can be implied that person 3 and 5 are not friends. *This operation is carried out in constant time O(1) since it only performs an 'and' operation.*

The ease of carrying out the three general operations mentioned above owes its gratitude to the way the proposed framework is structured. There is no effort needed to ensure that a sorted list is to be maintained for every change that happens in the framework. The changes are localized and it won't affect the other person's bit-strings. In effect, the changes to the framework only involves either setting or clearing of a pair of bits which are strictly of *O(1)*.

E. Collaborating two frameworks to make results more meaningful

Consider two frameworks, one storing the relations between people as shown in Fig(1) and the other framework storing the webpages named A, B, C, D, E, F, G that each person likes as shown in Fig(4).

	A	B	C	D	E	F	G
1st bit	1	1	1	0	1	1	0
2nd bit	0	1	0	1	1	0	1
3rd bit	1	0	1	1	0	1	1
4th bit	1	0	1	0	1	1	0
5th bit	0	0	1	0	0	0	1
6th bit	0	1	0	1	1	1	0
7th bit	1	1	0	1	0	1	1

Fig(4)

The bit-string under the webpage A is 1011001. Here, bits 1, 3, 4 and 7 are set. This indicates that persons 1, 3, 4 and 7 like the webpage A and persons 2, 5 and 6 do not like page A as their corresponding bits are not set. It is to be noted that this particular framework shown in Fig(4) includes information about two different entities: webpages and people. Also, this framework is not an adjacency matrix as it has captured information about two entities and hence the framework is not symmetric.

As described in Section IV. B, the same operations can be performed on the current framework. For example, Fig(5) shows the result of *binary and* operation that is performed on the bit-string under columns A and B.

	A	B	A and b
1st bit	1	1	1
2nd bit	0	1	0
3rd bit	1	0	0
4th bit	1	0	0
5th bit	0	0	0
6th bit	0	1	0
7th bit	1	1	1

Fig(5)

As stated in Section IV.B.1, the result of the *conjunction* operation indicates the commonalities between two sets. In this example, the 4th column of Fig(5) has the bit-string 1000001. This indicates that person 1 and 7 both like pages A and B. Since persons 1 and 7 like the same page, it can be inferred that they have a common area of interest. Referring to the framework shown in Fig(1), persons 1 and 7 are not friends with each other yet. Since persons 1 and 7 like the same page, they are likely to become friends if their names are suggested to each other.

The complexity of performing this operation is again within *O(n)* and hence is efficient as described in Section IV.D.

Similarly, other operations can be performed on this framework and different, meaningful results can be easily and efficiently obtained. For example, an *exclusive-or* operation that is performed between two or more columns on the framework shown in Fig(4) determines the level of inconsistency of the people liking those pages involved in the operation.

V. IMPROVEMENTS

The memory organization of the proposed framework can be improved with a small computational cost, not exceeding *O(n).* Fig(1) portrays an adjacency matrix which is always symmetric along the principal diagonal. It would be sufficient if we just store only one of the symmetric halves. This would result either in an *upper triangular matrix* as shown in Fig(6). We also need not store the principal diagonal elements as we know that they are always zero.

	1	2	3	4	5	6	7
1st bit	-	1	1	0	1	1	0
2nd bit			0	1	1	0	1
3rd bit				1	0	1	1
4th bit					1	1	0
5th bit						0	1
6th bit							0
7th bit							-

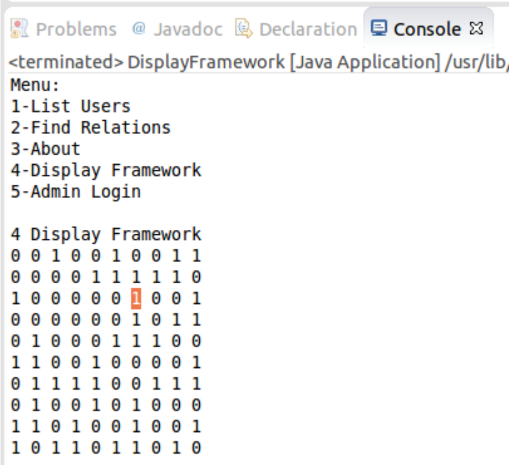
Fig(6)

Whenever we want to perform any operation on the framework, the other symmetric half only for that particular person can be constructed in *O(n)* and the desired operation can be carried out.

This improvement results in saving memory by 50% without exceeding the time complexity of O(n).

The framework can be segregated into components that are classified on a strong basis. For example, we can have separate frameworks for different countries, considering the likelihood of a person not befriending a person outside his/her country. The main idea is to reduce the value of *n.* Since most of the operations depend on *n*, The operations can run faster if *n* is small. So, if the framework consists of 1000 people, some way of classification can be made to reduce the number of people in each component to not more than 32.

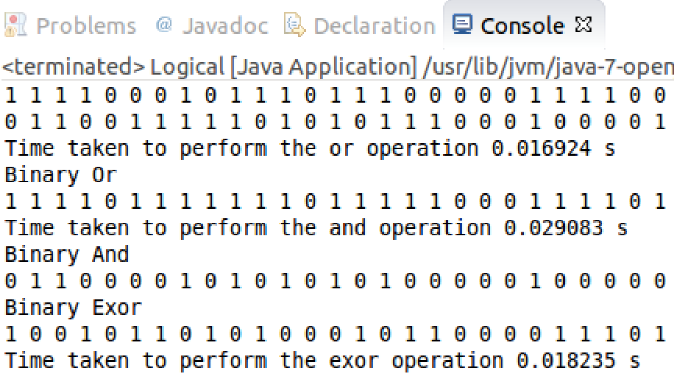
VI. RESULTS



```
<terminated> DisplayFramework [Java Application] /usr/lib,
Menu:
1-List Users
2-Find Relations
3-About
4-Display Framework
5-Admin Login

4 Display Framework
0 0 1 0 0 1 0 0 1 1
0 0 0 0 1 1 1 1 1 0
1 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 1 0 1 1
0 1 0 0 0 1 1 1 0 0
1 1 0 0 1 0 0 0 0 1
0 1 1 1 1 0 0 1 1 1
0 1 0 0 1 0 1 0 0 0
1 1 0 1 0 0 1 0 0 1
1 0 1 1 0 1 1 0 1 0
```

Fig(7)
Displaying the framework.



```
<terminated> Logical [Java Application] /usr/lib/jvm/java-7-open
1 1 1 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 0 0 1
Time taken to perform the or operation 0.016924 s
Binary Or
1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 1 0 1
Time taken to perform the and operation 0.029083 s
Binary And
0 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0
Binary Exor
1 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 0 1 1 1 0 1
Time taken to perform the exor operation 0.018235 s
```

Fig(8)

Displaying the time taken to perform the operations described in Section IV.B.

VII. Conclusion

The proposed framework serves as a very efficient and powerful model for storing related data. Any other model would require either one of the aspects - memory or efficiency to be compromised to obtain a gain. But the proposed framework has been successful in winning over both the aspects - memory efficiency and efficiency in computing as well (*time complexity*) as described in Section IV.

Apart from being efficient in storage and time, it is also successful in rendering the most relevant and meaningful information about the entities that are in the framework. The ability to combine two frameworks (like framework storing people and framework storing webpages) as described in Section IV.E proves how flexible the proposed framework is, and also depicts how relevant the search information would be.

Frameworks that use other models, (for example, a graph model) to represent related data might provide almost the same flexibility in representation but it takes a hit in being efficient as it may have to go with graph traversal algorithms whose time complexity turn out to be a combinatorial explosion (*O(n!)*). Moreover, the ability to retrieve relevant search results is not guaranteed in such frameworks.

The proposed framework is quite natural and simple. It does not make use of complex and arcane ways of representing data. It only uses an adjacency matrix for representation. The operations and search results are also simple and meaningful, and most of all, *relevant.*

References

[1] http://en.wikipedia.org/wiki/Data_management

[2] http://en.wikipedia.org/wiki/Data

[3] http://www.technologyreview.com/news/514351/has-big-data-made-anonymity-impossible/

[4] http://en.wikipedia.org/wiki/Raw_data

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[8] Social Referral: Leveraging network connections to deliver recommendations. Mohammad Amin, Baoshi Yan, Sripad Sriram, Anmol Bhasin and Christian Posse. To appear in Proceedings of the Sixth ACM conference on Recommender Systems (RecSys 2012)

[9] User Interests Identication on Twitter Using a Hierarchical Knowledge Base, Pavan Kapanipathi, Prateek Jain, Chitra Venkataramani, and Amit Sheth. Pregel: A System for Large-Scale Graph Processing

[10] Pregel: A System for Large-Scale Graph Processing,

Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski