e-Research: A Journal of Undergraduate Work

Volume 2
Number 2 *Vol 2, No 2 (2011)*

Article 1

September 2014

Full Issue

e-Research Editors

e-Research: A Journal of Undergraduate Work

Volume 2, Number 2 (2011)

Table of Contents

Introduction 37

Articles

Mapping Regions of RNF168 Required for its Degradation by ICP0 39
Andrea Cyr, Matthew Weitzman

High Performance Computing Markov Models using Hadoop MapReduce 45
Matthew Shaffer

A Two-Light Version of the Classical Hundred Prisoners and a Light Bulb Problem 57
Alexander Barrett, Cyril Rakovski

The Bernoulli Family 65
Dung Bui, Mohamed Allali

Utilizing Remote Sensing to Detect Deepwater Horizon Oil Spill Properties 77
Amanda Kristedja

About the Contributors 89

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **Introduction**

The 2011 issue of Chapman University's E-Research Journal samples research produced by undergraduates studying molecular genetics, modeling and simulations, and mathematics. Mr. Barrett used simulations to test increasingly complex strategies to solve a classical mathematical puzzle, one hundred prisoners and a light bulb. Ms. Cyr mapped the region of the herpes simplex virus essential to transition the virus from a latent to an active state. Ms. Kristedja used remote sensing to study the effects of Deep Water Horizon oil spill on chlorophyll concentrations in the Gulf of Mexico. Mr. Bui explored the contributions made by the Bernoulli family to mathematics and physics. Finally, Mr. Shaffer used the probability modeling tools, Markov Models and Hadoop MapReduce to determine the frequency with which specific words are followed by other specific words.

The undergraduate student research published here represents the dozens of research projects completed this year by Schmid College of Science and Technology students. Undergraduate research is a defining feature of Schmid College degree programs, thus, by the time students graduate from Chapman University they have completed at least one research project. We are proud that our undergraduates leave Chapman having had hands on experience undertaking the quest for new knowledge and for finding solutions to real world problems.

Janeen Hill, Guest Editor
Professor of Biological Sciences and Senior Associate Dean, Schmid College of Science and Technology

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

Mapping Regions of RNF168 Required for its Degradation by ICP0

Andrea Cyr, Matthew Weitzman*

ABSTRACT

Viruses establish infection by overtaking host cell processes and developing mechanisms that promote viral replication. Herpes simplex virus undergoes lytic and latent cycles of infection throughout the lifespan of its host. The viral genome is transcriptionally silent during latency, but viral proteins are produced upon reactivation. Herpes simplex virus type 1 encodes the ICP0 protein, an E3 ubiquitin ligase required for reactivation from latency of the infectious virus. The immediate-early protein ICP0 regulates the herpes simplex virus by activating viral gene expression thereby initiating lytic infection. Cellular proteins are degraded by ICP0, promoting the virus to enter the lytic cycle. Two cellular histone ubiquitin ligases, RNF8 and RNF168 that promote recruitment of DNA repair factors to DNA damage sites. ICP0 viral expression counteracts RNF8 and RNF168 as anti-viral factors by their degradation. This study maps regions of RNF168 required for its degradation so that proteins involved in viral reactivation can be determined. RNF168 has been shown to organize the DNA damage response by increasing the local concentration of ubiquitinated proteins in damage sites to prolong the ubiquitination signal. As a chromatin-associated RING finger protein, RNF168 consists of a sequence of two motif interacting with ubiquitin (MIU) domains that allow RNF168 to be recruited to damage sites. Two constructs were created, which separated each MIU to test for degradation by ICP0, thus localizing how ICP0 targets cellular proteins. In understanding how ICP0 targets its cellular substrates, other proteins targeted by ICP0 with similar motifs may be identified.

Keywords: Herpes simplex virus, ICP0 protein, Regulation of Herpes simplex virus, RNF 168

INTRODUCTION

Herpes Simplex Virus Type 1(HSV-1) is a double stranded DNA virus with a biphasic lifecycle. A vast majority of the U.S. population is seropositive and experience outbreaks of bothersome cold sores on the vermillion border of the lip (Lachmann, 2003). The virus causes primary infection by entering the lytic cycle within the epithelial cells in the skin of its human host. HSV-1 enters the sensory neurons and travels retrogradely via axons to the ganglia to establish latency of the viral genome, thus the relationship between the virus and these neurons are central to viral survival (Lachmann, 2003). In latency, all transcriptional expression of viral proteins is silent. During this state, only latency-associated transcripts (LATs) are expressed, and the function of these remains unclear (Everett, 2000). During ultraviolet exposure or stress, the virus spontaneously travels back through the neuron to re-establish infection in the epithelial cells, to cause recurrent lesions. The virus encodes immediate early (IE) genes, which aid the cells survival during lytic infection (Lachmann, 2003). IE genes are necessary for the activation of a later class of promoters (Everett, 2000). A complex is formed that requires the function of VP16, which uses the transcriptional machinery of the host cell and Oct-1, another cell protein. A third component, HCF, acts with VP16 and Oct-1 to strongly activate the IE gene expression (Everett, 2000). One IE gene in particular, ICP0, acts as a regulating switch between the lifecycles of the virus (Everett, 2000).

ICP0, an E3 ubiquitin ligase, is one of the first genes activated by the viral genome to regulate viral gene expression (Everett, 2000). ICP0 degrades specific proteins that aid viral DNA replication. Although it is non-essential to the virus, its absence negatively affects viral replication initially (Everett, 2000). The zinc-binding RING finger motif of

ICP0 is necessary for its function. Viral proteins interact with cellular proteins to regulate gene expression. Rather than binding to DNA directly, ICP0 acts by ubiquitin-proteasome pathways of cellular protein and nuclear domain structure degradation (Everett, 2000).

Once primary infection is established, the host cell recognizes the viral genome as DNA damage, thus a network of signaling pathways are coordinated by the cells DNA repair machinery (Lilley et al, 2010). Several factors are sent by the cell to respond to the invasive viral genome as recognized by the cell. A series of phosphorylations produce irradiation-induced foci (IRIF)-like subnuclear structures near the incoming viral genome. IRIF organization involved in marking sites of cellular DNA damage is still being studied (Huen and Chen, 2008). A mediator protein, Mdc1 (Mediator of DNA damage check point protein 1), interacts with gH2AX and ATM (Lou et al, 2006).

Two cellular ubiquitin ligases, RNF8 and RNF168, are important in the formation of IRIF (Huen et al. 2007). RNF8 binds to phosphorylated Mdc1 and ubiquinates H2A and H2AX histones (Huen et al 2007). The second ubiquitin ligase, RNF168, reinforces this mechanism by further modifying ubiquinated histones (Doil et al, 2009; Stewart et al, 2009). These mechanisms coordinate the recruitment of DNA repair proteins to sites of IRIF by proteins with ubiquitin interacting motifs and promoting chromatin restructuring (Huen et Chen, 2008). The parallels between IRIF and IRIF-like sub nuclear structures formed near incoming viral genome are a research focus.

In a previous study to determine a link between viral reactivation and damage signaling, ICP0 was found to degrade RNF8 and RNF168 (Lilley et al. 2010). Less DNA repair proteins accumulate at sites of DNA damage once ICP0 induces the degradation of RNF8 and RNF168 (Lilley et al. 2010). Proteasome-mediated degradation of RNF8 and RNF168 is ICP0-dependent (Lilley et al. 2010). In the absence of ICP0 (mutated viral genomes deleted for ICP0), RNF8 and RNF168 are not degraded. RNF8 and RNF168 interact with ubiquitin to target histone H2A and mark regions of DNA damage. Histone H2A directs DNA repair proteins to these sites. In the absence of ubiquinated H2A, the accumulation of repair proteins is hindered and viral genes are expressed. Degradation of RNF8 and RNF168 is responsible for the loss of H2A ubiquitination and disruption of IRIF in the presence of ICP0 (Lilley et al. 2010).

RNF168 is involved in mediating the recruitment of DNA repair proteins. RNF168 includes a RING finger motif and two motifs interacting with ubiquitin (MIUs) (amino acids 168-191 and 439-571). The RING finger motif (amino acids 15-58) of RNF168 interacts with other cellular factors to add ubiquitin onto target proteins. Mapping the regions required for RNF168 degradation by ICP0 gene expression may indicate a more precise location and a specific motif in RNF168.

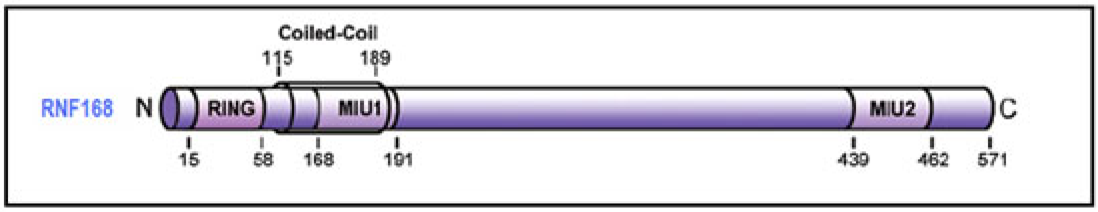


Figure 1: Schematic image of RNF168 containing a RING finger motif, MIU1, and MIU2.

MATERIALS AND METHODS

Cloning

PCR

Used Polymerase Chain Reaction for amplification of two RNF168 fragments. Ordered 2 forward and 2 reverse primer sequences from Eton. Primers prepared from 100 mM stock to 10 mM. Forward primers contain Flag sequence. 2X Mastermix containing dNTPs, polymerase and buffer was added to 100 ng Flag Riddlin DNA, and forward and reverse primers for either F1 or F2. Fragment 1 (F1) contain EcoRV and XhoI restriction sites and

Fragment 2 (F2) contain EcoRI and XhoI restriction sites. Cycler conditions set to 30 rounds of 95 degrees C for 1 min for initialization, 95 degrees C for 30 seconds to denature, 55 degrees C for 30 seconds to anneal, 72 degrees C for 1 minute for elongation, and 72 degrees C for 7 minutes for final elongation. Samples were run on an 8% agarose gel electrophoresis and stored at 4 degrees C. Followed QIAquick PCR Purification Kit protocol from QIAGEN using a micro centrifuge to purify amplified DNA.

Digests and Ligation

F1 digested with buffer 3, 10X BSA, EcoRV and XhoI restriction enzymes. F2 digested with buffer EcoRI, 10X BSA, EcoRI and XhoI restriction enzymes in 37 degrees C for at least 1 hour. Two vector digests using pcDNA 3.1 prepared for each fragment. 1:20 diluted CAIF Intestinal phosphatase added to vector digests. Digests run on an 8% agarose gel electrophoresis and purified using QIAquick Gel Extraction Kit protocol. F1 ligated with vector digest of EcoRV and F2 ligated with vector digest of EcoRI. Both ligations incorporated 50 ng vectors for 1:5 ratio of vector to insert.

Transformation and DNA Preparation

Competent TAMI bugs stored at -80 degrees C. 50 ul bugs to each ligation and incubated on ice for 30 minutes. Heat shock in 42 degree C water bath for 30 seconds. SOC media added to each ligation and incubated in 37 degree C water bath for 1 hour. Transformations plated on LB-Carb plates and incubated at 37 degrees C overnight. Colonies from F1 and F2 selected and incubated while shaking at 37 degrees C with LB and Amp. Plasmid DNA purified following protocol from QIAprep Miniprep protocol and QIAGEN Plasmid Maxi kit protocol.

Cell lines

HeLa and 293T cells were purchased from the American Tissue Culture Collection. Routine passage of adherent cell lines maintained using PBS buffer as wash and 10% trypsin: EDTA (10 ml trypsin solution in 100 ml versene solution) for cell detachment. Cells resuspended and maintained in Dulbecco modified Eagles medium (DMEM) containing 100 U/ml of penicillin and 100ug/ml of streptomycin, and supplemented with 10% foetal bovine serum (FBS) and antibiotics. Cells were grown at 37 degrees C in a humidified atmosphere containing 5% CO2. Cells were routinely split as most appropriate in separate 15 cm plates.

Transfections

Cells were subjected to Lipofectamine 2000 (LF 2000) transfection in 6-well plates. 40 ul LF 2000 master mix prepared with 2.5 ml optimem. Mixed with 4ug plasmid in a 3:1 ratio of ICP0 to RNF168 of fragment 1-12 (F1-12), fragment 1-15 (F1-15), or full-length RNF168 DNA in 250 ul optimem. Media on cells during transfections contained 10% FBS without penicillin and streptomycin. After 6 hours at 37 degrees C, media was replaced to contain antibiotics.

Whole Cell Lysates

Cells transferred in 1.5 mL eppy tubes and micro centrifuged at 4000g for 5 min. Pellet stored at -80 degrees C. Whole cell lysis buffer prepared (1M PBS, 1000mM Na Vanidate, 1M B-glycerol phosphate 50X, 1M NaF 50X, 1 crushed Complete +EDTA in 1 ml H2O 50X, 0.1mM PMSF, 10% NP40, 0.1% Triton) and added 40 ul per pellet. Incubated 20 minutes on ice and micro centrifuged 20 minutes at 4 degrees C. Supernatent stored at -80 degrees C.

Lowry Assay

6 standards (final concentrations: 0, 100, 200, 400, 600, 800 ug/ml) prepared in duplicate. Samples added with 125 ul mix of 6.25 mls Biorad protein assay reagent A and 125 ul reagent S. 1 mL Biorad protein assay reagent B added to each sample. Concentrations read on UV spectrometer.

Western Blotting

10-well and 12-well 4-12% Bis-Tris Acetate gels with MOPS running buffer to run 18 ul samples of 30 ug samples and 60 ug samples respectively and full range rainbow recombinant protein marker. 1:5 loading dye prepared with LDS sample buffer (4X) and IM DTT (20X). Gels ran at 150 V for 1.5 hours. Transfers ran at 30 V for 2 hours in 20X NUPAGE transfer buffer, methanol, H2O, NUPAGE antioxidant. Membranes soaked in 5% milk with 100 ul NaN3 after transfer in 4 degree C cold room. Membranes washed with 1X PBST. Membranes soaked with mouse ICP0, mouse GAPDH, or mouse Flag primary antibodies then horse-radish peroxidase-conjugated mouse secondary antibodies in 5% milk. Equal volumes of Western Lightening Plus-ECL oxidizing reagent and enhanced luminal reagent distributed over membranes for film development.

RESULTS

Cloning

Cloning the two separate constructs of RNF168 was carried out as described in the methods section. Correct constructs of the clones were identified by restriction digest analysis and computer-based DNA sequencing. The verified clones of F1 and F2 were purified and used in transfection experiments in the HeLa cells. As seen in figure 2, the F1 construct (~850bp) contained the RING finger motif and MIU1, and the F2 construct (~1Kbp) contained MIU2.

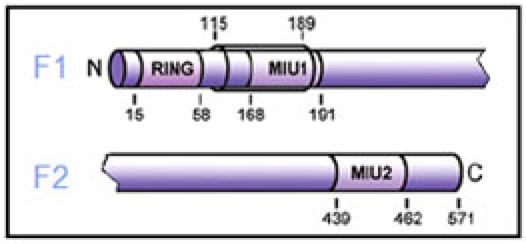


Figure 2: Schematic images of F1 and F2 constructs of RNF168 cloned and transfected +/- ICP0.

F2 is degraded by ICP0 expression

ICP0 expression degrades several cellular substrates, but this study focused on RNF168 degradation specifically. Degradations of the fragments from the separated RNF168 were compared to each other and to the full-length RNF168. The lysates of ICP0 co-transfections in HeLa cells were analyzed by a western blot for F2. F2 fragment degradation is evident with ICP0 expression as seen in Figure 3. The F2 fragment is not degraded with pcDNA3.1 vector alone in the absence of ICP0. F2 is sized at approximately 50kDa.

F1 may be degraded by ICP0 expression less efficiently

The western blot in Figure 3 indicates moderate degradation of F1 by ICP0 expression. Two separate lysates of F1 (F1-12 and F1-15) were analyzed on the western blot to express F1 in duplicate due to uncertainty of the quality of the sequences of the clones. F1 degradation appears to be less efficient in comparison to F2 degradation. Both F1-12 and F1-15 fragments suggest no degradation in the presence of pcDNA3.1 vector alone. F1 is sized at approximately 38kDa.

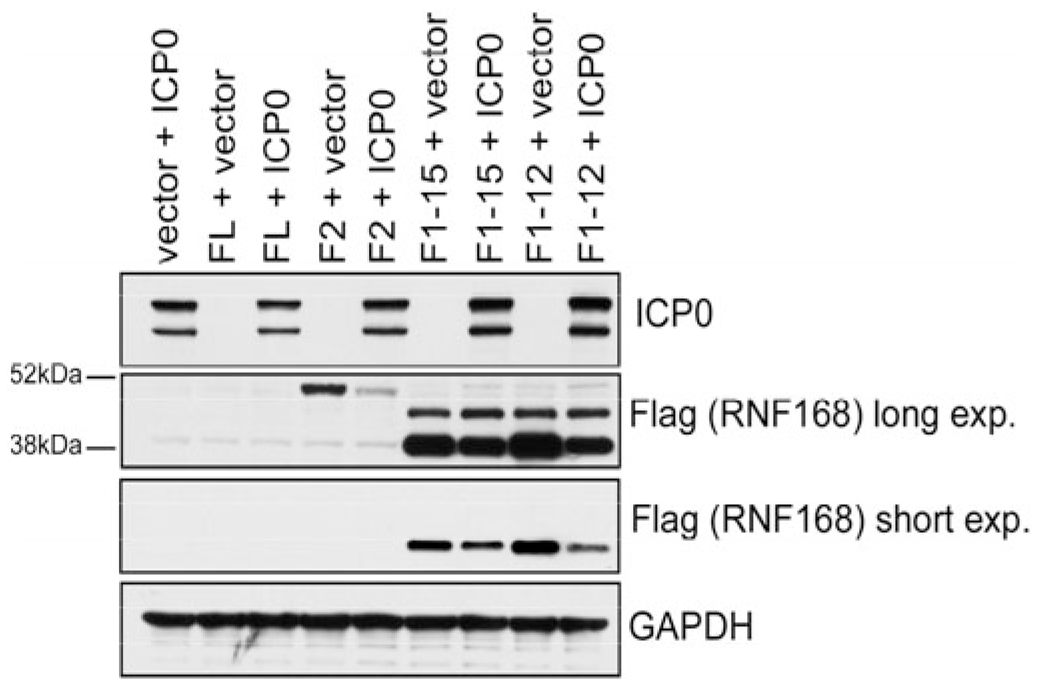


Figure 3: ICP0 expression results in Fragment 2 (F2) degradation at ~50kDa. ICP0 expression results in possible moderate degradation of Fragment 1 (F1) at ~38kDa with blots exposed for 1 min and 3 seconds. HeLa cells were co-transfected with ICP0 and harvested for 24 hrs post transfection. Lysates were analyzed by western blotting and assessed for levels of F1 or F2. Full length RNF168 controls show no expression, likely due to experimental error. GAPDH was used as a loading control. Flag-tag was added onto all RNF168 fragment clones for detection.

DISCUSSION

Based on previous studies, RNF168 is a target for ICP0-mediated degradation (Lilley et al. 2010). This study constructed fragments of RNF168 by truncating the substrate into two fragments to determine their degradation by ICP0. F1 was truncated to contain both the RING finger domain and MIU1. F2s prominent feature was the MIU2. By mapping the region of degradation, the target location on RNF168 can be better determined. Due to the fact that F2 is effectively degraded, it has led us to believe that the target location of RNF168 is found within F2. MIU2 may potentially be the target location for ICP0-mediated degradation, yet further truncations need to be analyzed before reaching more specific conclusions.

Figure 3 indicates F1 degradation with ICP0 expression, but less efficiently than F2, which is clearly degraded. Since both fragments are degraded to some degree, other interacting factors involved in substrate degradation could be present ICP0 may not directly target RNF168, but other factors are involved by interacting on multiple locations on the RNF168 protein. F1 fragment lysates were tested in duplicate because their quality, as determined by computer-based DNA sequence analysis, was questionable. Both F1 fragments, however, appear to have been successfully transfected and show very similar results.

No bands were found on the full length RNF168 fragment on the western blot as seen in Figure 3. Based on previous studies, full length RNF168 is degraded in the presence of ICP0 expression. The lack of bands for this control is attributed to experimental error. Possible explanations include that the full-length RNF168 construct may not have contained the flag sequence necessary for antibody detection, or an erroneous transfection occurred.

A separate study investigated a mutated RNF168 protein in the RIDDLE syndrome (Stewart et al, 2009). This severe immunodeficiency syndrome, from altered development of the immune system, results in dimorphic features and learning difficulties. Two different RNF168 truncations of the protein affect repair protein accumulation at DNA damage sites. Both truncations lack MIU2 indicating its possible necessity for full RNF168 function. Those specific studies suggest that both Ubiquitin binding and ligase activity are necessary for full RNF168 function. Partial RNF168 activity in human syndromes such as in the RIDDLE patient may allow only certain cellular processes to proceed. Such clinical cases could benefit from improved understanding of the protein degradation mechanism of RNF168.

After mapping regions of RNF168, further investigations should research a mutant virus, which cannot degrade RNF168, and how the substrates absence would affect viral replication and its lifecycle. Based on how ICP0 targets cellular substrates that are likely important in transcriptionally silencing the viral genome, other proteins targeted by ICP0 with similar motifs could be identified. Once the location that ICP0 interacts with the cellular substrate RNF168 is determined, future studies would include mapping the region responsible for degradation on the viral protein ICP0.

REFERENCES

Doil C, Mailand N, Bekker-Jensen S, Menard P, Larsen DH, Pepperkok R, Ellenberg J, Panier S, Durocher D, Bartek J, Lukas J, Lukas C (2009) RNF168 binds and amplifies ubiquitin conjugates on damaged chromosomes to allow accumulation of repair proteins. *Cell* 136: 435446

Efstathiou S, Preston CM (2005) Towards an understanding of the molecular basis of herpes simplex virus latency. *Virus Res* 111: 108-109

Everett RD (2000) ICP0, a regulator of herpes simplex virus during lytic and latent infection. *Bioessays* 22: 761770

Huen MS, Chen J (2008) The DNA damage response pathways: at the crossroad of protein modifications. *Cell Res* 18: 816

Huen MS, Grant R, Manke I, Minn K, Yu X, Yaffe MB, Chen J (2007) RNF8 transduces the DNA-damage signal via histone ubiquitylation and checkpoint protein assembly. *Cell* 131: 901914

Lachmann R (2003) Herpes simplex virus latency. *Expert Reviews* Vol.5; 5.

Lilley C, et al. (2010) A viral E3 ligase targets RNF8 and RNF168 to control histone ubiquitination and DNA damage responses. *EMBO Journal* 29: 943-955

Lou Z, Minter-Dykhouse K, Franco S, Gostissa M, Rivera MA, Celeste A, Manis JP, van Deursen J, Nussenzweig A, Paull TT, Alt FW, Chen J (2006) MDC1 maintains genomic stability by participating in the amplification of ATM-dependent DNA damage signals. *Mol Cell* 21: 187200

Stewart GS, Panier S, Townsend K, Al-Hakim AK, Kolas NK, Miller ES, Nakada S, Ylanko J, Olivarius S, Mendez M, Oldreive C, Wildenhain J, Tagliaferro A, Pelletier L, Taubenheim N, Durandy A, Byrd PJ, Stankovic T, Taylor AM, Durocher D (2009) The RIDDLE syndrome protein mediates a ubiquitin-dependent signaling cascade at sites of DNA damage. *Cell* 136: 420434

ACKNOWLEDGEMENTS

Thank you to Mira Chaurushiya and Caroline Lilley for mentoring me throughout the summer research program, The Salk Institute, the entire Weitzman Laboratory, namely Amy Blount for her cloning expertise, and the University of California-San Diego for welcoming my presence and a chance to explore my scientific opportunities, Amgen National Foundation for their gracious funding, Amgen Scholars Program and Magda Gramada for program coordination.

* Matthew Weitzman, Ph.D., Laboratory of Genetics, The Salk Institute for Biological Studies, La Jolla, CA, USA

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **Shaffer**

High Performance Computing Markov Models using Hadoop MapReduce

Matthew Shaffer

Abstract

In this paper, I will explain how I used the probability modeling tool, Markov Models, in combination with Hadoop MapReduce parallel programming platform in order to quickly and efficiently analyses documents and create a probability model of them. I will explain what Markov Models are, give a brief overview of what MapReduce is, explain why Markov models can be used for document analysis, explain my code of the modeling program, and examine the performance of various MapReduce platforms and techniques in analyzing documents.

Keywords: Probability Modeling Tools, Markov Models, Hadoop Maproduce, Cloud Computing

1 Introduction

For this experiment, Hadoop MapReduce was used to parallelize the process of creating Markov models. Markov Models are a widely used probability modeling tool. They model any environment that can be separated into discrete "states" where the probability of moving from one state to another depends only on the current state. Hadoop's MapReduce framework is an open source programming library that uses the techniques introduced by Google's MapReduce process in order to program computers to store and process vast amounts of data efficiently.

In this project, a program was encoded to analyses documents into a Markov model by modeling the probability of any particular word following another word. The local Chapman Hadoop node along with the Amazon Web Service Cloud computing platform were used in order to test how various MapReduce platforms perform. The results of these experiments were analyzed to give a rough overview of the efficiency of the various programs in processing data.

2 Parallel Programming Environment Analysis

MapReduce is a software framework first introduced by Google and designed to process large amounts of data by separating the data into smaller chunks and performing large numbers of small operations in parallel on the data. It has two steps. The first is to divide a data set up into groups and input into the "Map" operation. This operation works by taking the input data and output a set of "Key-Value" pairs. These pairs are generally a set of tuples in no particular order that are outputted as they are found. The next step is to organize the data by sorting the keys and aggregating the values together. This data may then be given to another map step, in the case of a multi-step MapReduce job, or then given to the "reduce" step. This step works by taking all the values for a key and summing them up into a result which is then outputted from the program.

The Hadoop MapReduce Framework is an open source version of MapReduce developed by the Apache Software Foundation[5]. The Hadoop MapReduce Framework offers two main ways of running MapReduce program: Streaming and Jar files. The streaming system works by launching an outside program and feeding in the data to the program as the standard in and reading the results from the standard out. The Data is "streamed" in and out of another program. The jar file implementation works by taking a jar file that contains a program that extends the MapReduce base and running it using the standard MapReduce API.

Streaming functions differently from the Jar file functionality in how information is moved from the "map" step to the "reduce" step. A Jar file program will sum up data into an aggregate set of data between these two steps. For Example: <Book, 1>, <Duck, 1>, <Book, 1> becomes <Book, 1 1>, <Dog, 1>. The Streaming operation simply organizes the keys but does not aggregate them. For example, the above would become: <Book, 1>, <Book, 1>, <Duck, 1>.

The Hadoop framework allows for any of its serializable datatypes to be used as the key and values in a jar file job. For example, in the program used for this paper Text objects were used as both the input and output of each step. This makes the actual job more flexible and able to process more types of programs and data.

2.1 Markov Models A brief analysis

Markov Models are a probability modeling tool. A Markov Models is any model, such as a graph, that has the Markov Property. The property applies to any system such that the system has discrete states and given the present state of the system and all its previous states the probability of the state changing from the current one to another depends only on the current state. In the case of discrete values and time indexes,

$$\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1} \ldots X_0 = x_0) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})$$ [1].

This means that at any time the probability of anything happening depends only on the present situation and not on any previous situation. This allows a model to show probability as a simple statistically likelihood of the transition between states while disregarding pervious state. This simplifies the task of predicting the likelihood of something happening by making it so that only a single model and its probabilities need to be considered at any time.

For example, for my project uses Markov models to model documents. A document can be viewed as one word following another many times over. A Markov model can be made to model this behavior by examining how often one word changes to another word. Thus each word is a state and the models finds how likely a state is to change to another state, or word here, based on the current word and not on any previous states, or words.

Often times a technique called "smoothing" is used on Markov models. This technique gives the model at least a small probability of transitioning from any one state to any other state regardless of whether or not a particular state followed another in the data set. For example, there would be a small chance for elephant to follow the word river even if the word elephant never followed river in any of the data that was used to make the model. This technique helps to simulate the randomness of an actual environment and to try and deal with situations where an event can occur but simply never did in the data set that was used to generate the model.

3 Parallel Computing Experiments

MapReduce is particularly suited for Markov modeling. Since the probability of moving to each state depends only on the current one there is no need to record what words precede the current one. This means that the data can be easily divided up into small chunks without losing any information. In addition, the idea of key-value pairs works well with the Document analysis since each word can be broken up into word-next word pairs that follow the key-value method and then the number of times a word follows another can be summed in the reduce step in the same way other aggregated values are summed up in reduce.

My Python and Java Programs are based on the example WordCount programs provided by Hadoop on their website [2]. I will explain my Python code and then explain my Java code.

MarkovMapper:
The inputted lines (from Standard in) are formatted by removing punctuation, converted the string to lower case, and removing extra white space. They are then moved into a list of individual words. This formatting is to avoid creating erroneous distinctions between capitalized letters and uncapitalize letters and words with, for example, a period at the end and those that don't have a period. This makes the words more uniform and the results more accurate.

```
for line in sys.stdin:
   line = line.strip().lower().translate(None, string.punctuation)
   words = line.split()
```

Then, the words, assuming that there are any at all, are printed (to standard out) in order as the word and then the next word in the list after it. A tab is used to distinguish the key from the value.

```
  if(len(words) > 0):
          for i in range(len(words)-1):
        print '%s\t%s' % (words[i], words[i+1])
```

MarkovReducer:
As words are read in (from Standard in), they are split using the tab that was used to distinguish keys and values in the previous step. The key-value pair is then checked against the contents of a dictionary in the program, called counter. If the key does not already exist, it is added with a value of 1 otherwise the previous value is incremented.

```
for line in sys.stdin:
   nkey, word = line.split('\t', 1)
   if (nkey, word) not in counter:
      counter[(nkey, word)] = 1
   else:
      counter[(nkey, word)] += 1
```

After each line has been inputted, the keys of the dictionary, i.e. the key-value tuple, are collected into a list. The key, followed by the value and the contents of the key-value in the dictionary, i.e. the counter for this pair, is printed with it.

```
outwords = counter.keys()
for outword in outwords:
   output += str(outword[0]) + "\t" +str(outword[1])+ " " + str(counter[outword])+
"\n";
print output
```

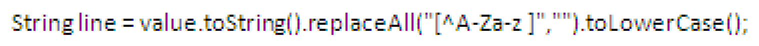
Thus, the mapper simply creates tuples of words and the word that follows them and the reducer sums up the number of each pair by using a dictionary and prints out the counts for each pair.

Markov.java:

Note: Both my Map and Reduce classes have been altered from using the default key-value pairs of text and integer to text and text by changing the map and reduce functions like so:

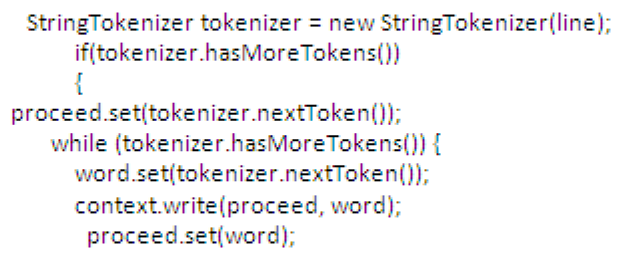
```
public static class Map
     extends Mapper<LongWritable, Text, Text, Text>{..
public static class Reduce
     extends Reducer<Text, Text, Text, Text> {...
```

My Mapper Function works by first taking each line that is inputted and stripping it of all characters except Letters and spaces and converting it lower case. Once again, this is to remove any apparent differences to the program between words that have varying cases and formats in order to make the words uniform and improve the test results.



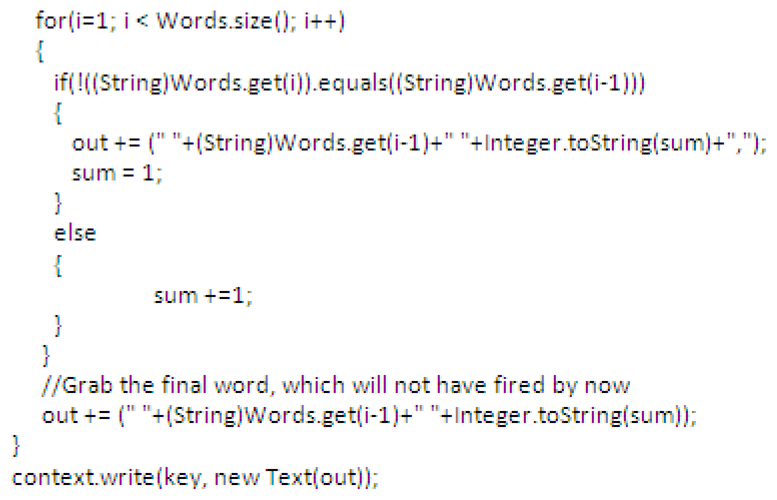
```
String line = value.toString().replaceAll("[^A-Za-z ]","").toLowerCase();
```

The line is then split into individual strings using the StringTokenizer. If there are any lines, the Tokenizer sets the initial word into a variable called previous. It then goes through each word and sets the word it reads to be the variable "word" and prints the word along with the previous word then making the current word the previous word. In this way, it prints each word along with the word that proceed it by simply going through each word and printing the old word with the current one then moving the current one into the previous spot.



```
  StringTokenizer tokenizer = new StringTokenizer(line);
        if(tokenizer.hasMoreTokens())
        {
proceed.set(tokenizer.nextToken());
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        context.write(proceed, word);
          proceed.set(word);
```

For the reducer, the program first creates an ArrayList and goes through each inputted value adding them to the Arraylist (as a String object). It then sorts it using Collections.sort(). The program then checks for the special case of only having one value in the list of words and simply creates an output of the word and the number one for this case. Otherwise, for each word, starting from index 1 it checks the word against the previous word. If the words are the same, a counter is incremented, if they are different then the word and the counter are added to an output string and the counter is reset. Once each word has been checked, the last word is added to the string (Since it will not have yet have been printed due to the fact that only different value words are printed and the last word cannot be checked as different since no words follow it). The Key along with its output string is then printed.



```
    for(i=1; i < Words.size(); i++)
    {
      if(!((String)Words.get(i)).equals((String)Words.get(i-1)))
      {
        out += (" "+(String)Words.get(i-1)+" "+Integer.toString(sum)+",");
        sum = 1;
      }
      else
      {
                  sum +=1;
      }
    }
    //Grab the final word, which will not have fired by now
    out += (" "+(String)Words.get(i-1)+" "+Integer.toString(sum));
}
context.write(key, new Text(out));
```

A Sample output for the Streaming program looks like this:

```
absorbing piece 1
absorbing some 1
absorbs all 1
absorbs into 1
absorbs it 1
absorbs light 1
absorbs my 2
absorbs the 3
absorbs war 1
```

```
absorption of 4
absorption the 1
absorption was 1
absorptive wall 1
abstain from 2
abstainer and 1
abstemious to 1
```

Each word is matched with a value and the number of times it appears is listed after that.

A Sample output for the Java code looks like:

```
abridged copy 1 1
abridgements all 1 1
abridges the 1 1
abridging the 1 1
abroad and 1, waylaying 1 1
abrupt and 1 1
abruptly back 1, bent 1, his 1, i 1, the 1, what 1 1
abscission of 1 1
absconded somewhere 1 1
absence and 1, of 7, on 1, through 1 1
absent face 1, others 1 1
absentee the 1 1
absently ocularly 1 1
absentminded beggar 3, beggars 1 1
```

Each word is followed by the list of words that followed it along with the number of times each followed it. Unfortunately, I was not able to remove the default formatting of adding a tab and a 1 in the final output for each value. I did not have the time or expertise to find where these defaults are specified in the code and change them.

4 Performance Analysis

To test the performance of my program I tested the speed of my program with single instance set-ups for both streaming python and the custom jar on the local server and for the Custom Jar on AWS. I also tested the speed up of the Jar File on increasing numbers of instances on AWS. I used 4 data sets to test performance: a small data set of 3546 KB, a medium data set of 14400 KB, and large data set of 35248 KB, and an extra large dataset of 55712 KB (all files were obtained from the Guttenberg Project website [3]).

To test on the local server, I used the "time" command while running each program to get a measure of the total time the program ran. I ran each ten times and average the results to get the final times.

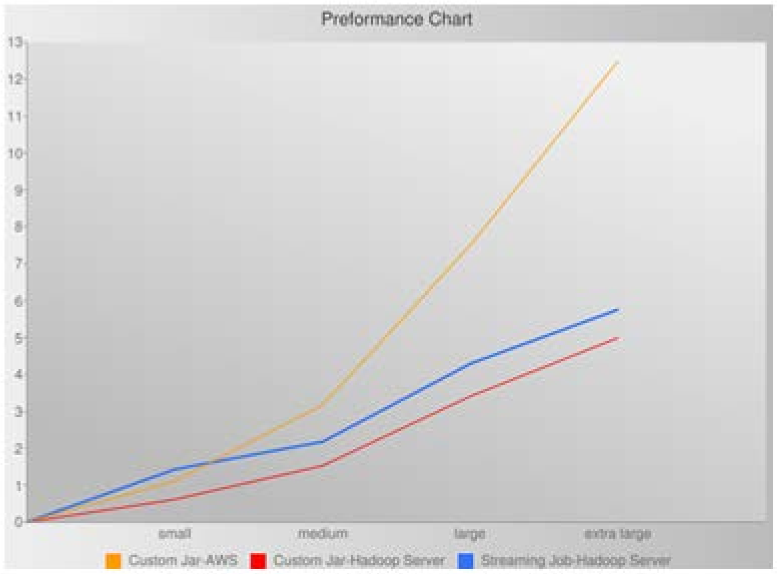
The Results are:
Streaming Python -
Small -1:26.378
Medium - 2:11.986
Large - 4:18.141
Extra Large - 5:46.468

Jar
Small - 0:37.112
Medium - 1:32.873
Large - 3:25.36
Extra Large - 4:59.918

The same data was uploaded to the Amazon Web Services, AWS, and the JAR file program was run on each dataset. Each dataset was run and recorded only 3 or 4 times due to the cost of running programs using AWS. The execution time was measured by comparing the time stamp of the first and last entry in the SysLog file of the job. Unfortunately, the streaming job fails due to unknown reasons during the shutdown phase so I was unable to get results for this type of job on the AWS.

The results are:
Small -1:07
Medium - 3:11
Large - 7:30
Extra Large - 12:30.5

I compiled the results into a chart that graphs time in minutes against data size:



The Data set sizes do not conform to a linear scale up, but, even so, it is clear from my results that on average the Jar file preformed better then the streaming job. This may be due to the overhead of the streaming of information from and to the python processes. The AWS was also fairly slow but this is likely due to the time it takes to initialize some functions on the cloud as well as the possibility that a single cloud instance has less computing power or memory then the local server.

As a further experiment, a speed-up test was run. Speed-Up is measured as the time for the data to be executed on one core divided by the time it takes for the data to be executed on *p* cores. This was tested by taking the Extra Large data set and running it on the AWS with increasing numbers of instances. Only one test result was measured for each, due to the fact that running multiple tests with large number of instance could quickly become expensive. As a result, this data is only a relative indicator.

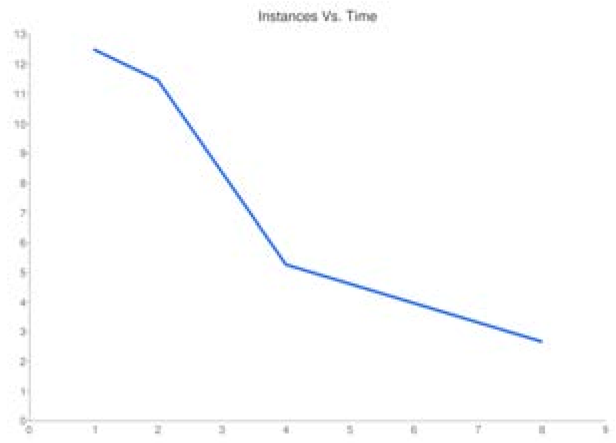
The results are:
1 instance -12:30.5
SpeedUp: 1

2 instance - 11:28
SpeedUp: 1.0908

4 instance - 5:16
SpeedUp: 2.375

8 instance - 2:40
SpeedUp: 4.6906

This chart plots the time it takes for the code to run vs. the number of instances:



In the speed-up test, the program appears to perform at roughly half of the ideal speed up. In ideal speed-up, two processors working twice as fast as one, four working four times as fast as one, and so on. This data is only a relative indicator, but points to a good speed-up but definitely one that could still be improved. Further testing and different datasets might reveal different results but this data does indicate the increasing benefit of using additional instances even if they do not perform ideally.

5 Conclusions

It is clear from my results that the Amazon web services preformed much slower than the local cluster. It is also clear that Hadoop continues to run well even when dealing with increasingly large datasets. Custom JAR files also appear to run faster overall then streaming jobs. Increasing number of instances also appear to, although not offering perfect speed up, do offer good speed up and provide a good benefit to using more processors on a job.

The process of streaming information in and out of a program seems to be a draw-back on performance speed. The transfer of information creates an additional overhead that detracts from the main process of processing the data. JAR files are thus the most effective way of using Hadoop. The speed-up performance of the system is also good but not ideal. Ideal performance is rare in any system and Hadoop does use additional cores effectively without significant loss of processing power when additional cores are added even at only half speed-up.

The AWS, although offering slower speeds here, might be hinder by potentially being slower than the local cluster overall and may be less fine-tuned to this particular type of computation. It is possible that the AWS which is a more general computing environment that that has a MapReduce environment layered onto it and must deal with sharing resources among multiple cloud users simultaneously is far more strained and has more overhead then a local cluster. This makes using a local system more efficient. The AWS is easier to use though since no equipment needs to be purchased, no system needs to be set up, and no maintenance is necessary for it. The AWS is simply an upload and go system that requires no upfront set up like a local computing cluster.

Overall, Hadoop does seem like a very efficient way to manage large datasets as even the 55 MB file was able to be processed in under 5 minutes on average with the simple 2 processor local cluster. This makes processing vast amounts of data quicker than the conventional system which is the equivalent of the much slower one instance system in this experiment.

Further research can be made into this field by exploring the speed-up of MapReduce in other types of programs, as well as exploring the speed-up of MapReduce Streaming jobs on AWS. A comparison of both types of jobs on the AWS and local server would yield interesting data into the general efficient of using the AWS cloud system as opposed to a local host. More research can also be done into exploring the optimal number of processes to work on a job by exploring speed-up vs Data size and finding the most efficient number of processes per unit of data and exploring to see if the AWS is more efficient at certain types of computations or programs then it is at other types of computations.

References

1. http://en.wikipedia.org/wiki/Markov_property

2. http://hadoop.apache.org/common/docs/r0.20.2/mapred_tutorial.html

3. http://www.gutenberg.org

4. http://en.wikipedia.org/wiki/Markov_chain

5. http://hadoop.apache.org/mapreduce/

6. http://hadoop.apache.org/common/docs/r0.20.2/mapred_tutorial.html

7. http://www.velocityreviews.com/forums/t138330-remove-punctuation-from-string.html

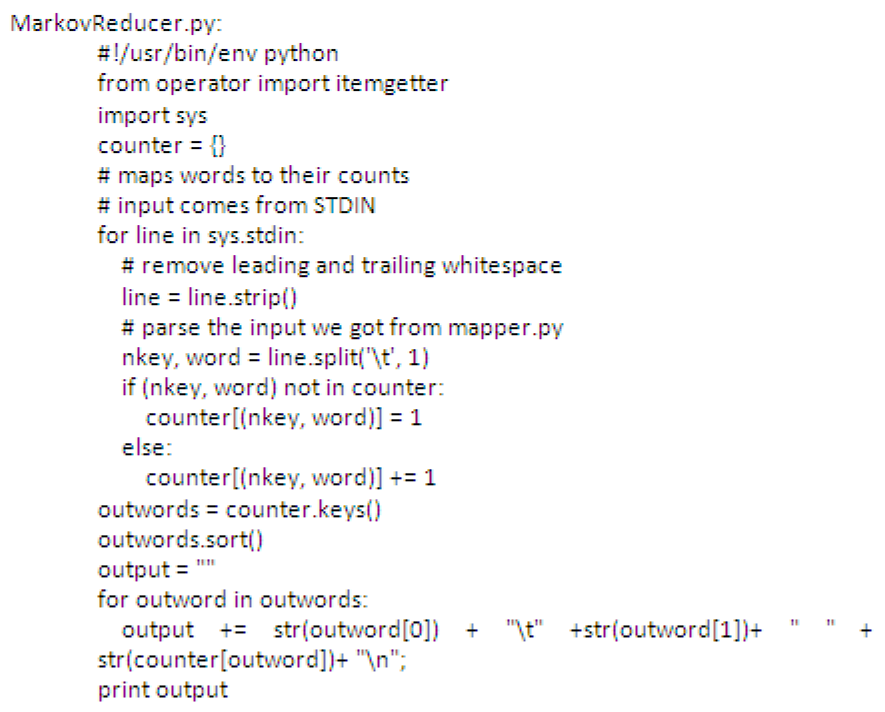
8. http://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string-in-python

9. http://hadoop.apache.org/common/docs/r0.18.3/streaming.html

Appendix

MarkovMapper.py:

```
#!/usr/bin/env python
import sys
import string
# input comes from STDIN (standard input)
for line in sys.stdin:
   # remove leading and trailing whitespace
   line = line.strip().lower().translate(None, string.punctuation)
   # split the line into words
   words = line.split()
   if(len(words) > 0):
           for i in range(len(words)-1):
      # write the results to STDOUT (standard output);
      # what we output here will be the input for the
      # Reduce step, i.e. the input for reducer.py
      #
      # tab-delimited; the trivial word count is 1
      print '%s\t%s' % (words[i], words[i+1])
```

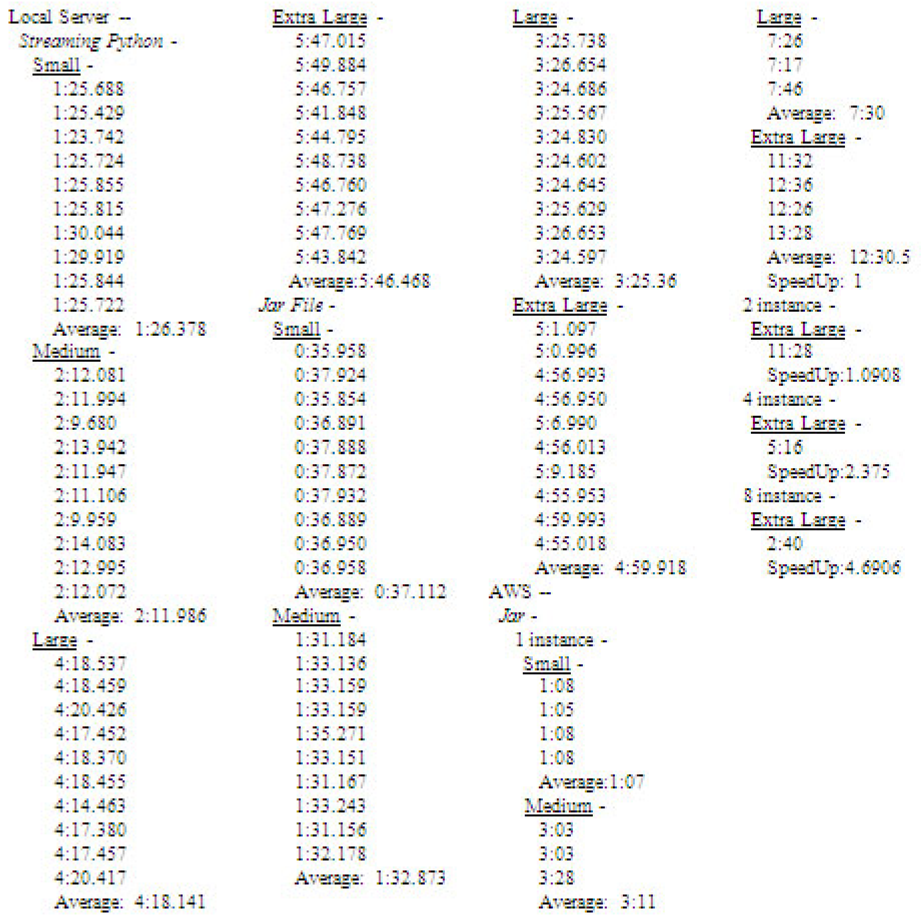


```
MarkovReducer.py:
        #!/usr/bin/env python
        from operator import itemgetter
        import sys
        counter = {}
        # maps words to their counts
        # input comes from STDIN
        for line in sys.stdin:
          # remove leading and trailing whitespace
          line = line.strip()
          # parse the input we got from mapper.py
          nkey, word = line.split('\t', 1)
          if (nkey, word) not in counter:
            counter[(nkey, word)] = 1
          else:
            counter[(nkey, word)] += 1
        outwords = counter.keys()
        outwords.sort()
        output = ""
        for outword in outwords:
          output    +=    str(outword[0])    +    "\t"    +str(outword[1])+    "    "    +
        str(counter[outword])+ "\n";
        print output
```

```
Markov.java:
        import java.io.IOException;
        import java.util.*;
        import org.apache.hadoop.fs.Path;
        import org.apache.hadoop.conf.*;
        import org.apache.hadoop.io.*;
         import org.apache.hadoop.mapreduce.*;
        import org.apache.hadoop.mapreduce.lib.input.*;
        import org.apache.hadoop.mapreduce.lib.output.*;
        import org.apache.hadoop.util.*;
        public class Markov extends Configured implements Tool {
          public static class Map
            extends Mapper<LongWritable, Text, Text, Text> {
           private Text proceed =new Text();
           private Text word = new Text();
           public void map(LongWritable key, Text value, Context context)
             throws IOException, InterruptedException {
            String line = value.toString().replaceAll("[^A-Za-z ]","").toLowerCase();
            StringTokenizer tokenizer = new StringTokenizer(line);
                 if(tokenizer.hasMoreTokens())
                 {
          proceed.set(tokenizer.nextToken());
               while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
           if(!word.toString().equals("") && !proceed.toString().equals("")){
                context.write(proceed, word);
                     proceed.set(word);
           }
              }
                 }
```

```
      }
    }
    public static class Reduce
        extends Reducer<Text, Text, Text, Text> {
      public void reduce(Text key, Iterable<Text> values,
          Context context) throws IOException, InterruptedException {
ArrayList Words = new ArrayList();
        for (Text  val : values)
{
                Words.add(val.toString());
}
Collections.sort(Words);
int sum = 1;
int i;
String out = "";
        if(Words.size() == 1)//if we only have one word, just write it
    out = ((String)Words.get(0)+" "+Integer.toString(1));
else
        {
    for(i=1; i < Words.size(); i++)
    {
      if(!((String)Words.get(i)).equals((String)Words.get(i-1)))
      {
        out += (" "+(String)Words.get(i-1)+" "+Integer.toString(sum)+",");
        sum = 1;
      }
      else
      {
                sum +=1;
      }
     }
     //Grab the final word, which will not have fired by now
     out += (" "+(String)Words.get(i-1)+" "+Integer.toString(sum));
}
context.write(key, new Text(out));
    }
  }
    public int run(String [] args) throws Exception {
      Job job = new Job(getConf());
      job.setJarByClass(Markov.class);
      job.setJobName("MarkovChain");
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(Text.class);
      job.setMapperClass(Map.class);
      job.setCombinerClass(Reduce.class);
      job.setReducerClass(Reduce.class);
      job.setInputFormatClass(TextInputFormat.class);
      job.setOutputFormatClass(TextOutputFormat.class);
      FileInputFormat.setInputPaths(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      boolean success = job.waitForCompletion(true);
      return success ? 0 : 1;
    }
    public static void main(String[] args) throws Exception {
      int ret = ToolRunner.run(new Markov(), args);
      System.exit(ret);
    }
  }
```

Raw Test Results:



```
Local Server --
 Streaming Python -
  Small -
    1:25.688
    1:25.429
    1:23.742
    1:25.724
    1:25.855
    1:25.815
    1:30.044
    1:29.919
    1:25.844
    1:25.722
    Average:  1:26.378
  Medium -
    2:12.081
    2:11.994
    2:9.680
    2:13.942
    2:11.947
    2:11.106
    2:9.959
    2:14.083
    2:12.995
    2:12.072
    Average:  2:11.986
  Large -
    4:18.537
    4:18.459
    4:20.426
    4:17.452
    4:18.370
    4:18.455
    4:14.463
    4:17.380
    4:17.457
    4:20.417
    Average:  4:18.141
  Extra Large -
    5:47.015
    5:49.884
    5:46.757
    5:41.848
    5:44.795
    5:48.738
    5:46.760
    5:47.276
    5:47.769
    5:43.842
    Average:5:46.468
 Jar File -
  Small -
    0:35.958
    0:37.924
    0:35.854
    0:36.891
    0:37.888
    0:37.872
    0:37.932
    0:36.889
    0:36.950
    0:36.958
    Average:  0:37.112
  Medium -
    1:31.184
    1:33.136
    1:33.159
    1:33.159
    1:35.271
    1:33.151
    1:31.167
    1:33.243
    1:31.156
    1:32.178
    Average:  1:32.873
  Large -
    3:25.738
    3:26.654
    3:24.686
    3:25.567
    3:24.830
    3:24.602
    3:24.645
    3:25.629
    3:26.653
    3:24.597
    Average:  3:25.36
  Extra Large -
    5:1.097
    5:0.996
    4:56.993
    4:56.950
    5:6.990
    4:56.013
    5:9.185
    4:55.953
    4:59.993
    4:55.018
    Average:  4:59.918
AWS --
 Jar -
  1 instance -
   Small -
    1:08
    1:05
    1:08
    1:08
    Average:1:07
   Medium -
    3:03
    3:03
    3:28
    Average:  3:11
   Large -
    7:26
    7:17
    7:46
    Average:  7:30
   Extra Large -
    11:32
    12:36
    12:26
    13:28
    Average:  12:30.5
    SpeedUp:  1
  2 instance -
   Extra Large -
    11:28
    SpeedUp:1.0908
  4 instance -
   Extra Large -
    5:16
    SpeedUp:2.375
  8 instance -
   Extra Large -
    2:40
    SpeedUp:4.6906
```

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **Barrett**

A Two-Light Version of the Classical Hundred Prisoners and a Light Bulb Problem: Optimizing Experimental Design through Simulations

Alexander S. Barrett, Cyril S. Rakovski

Abstract

We propose five original strategies of successively increasing complexity and efficiency that address a novel version of a classical mathematical problem that, in essence, focuses on the determination of an optimal protocol for exchanging limited amounts of information among a group of subjects with various prerogatives. The inherent intricacy of the problemsolving protocols eliminates the possibility to attain an analytical solution. Therefore, we implemented a large-scale simulation study to exhaustively search through an extensive list of competing algorithms associated with the above-mentioned 5 generally defined protocols. Our results show that the consecutive improvements in the average amount of time necessary for the strategy-specific problem-solving completion over the previous simpler and less advantageously structured designs were 18, 30, 12, and 9% respectively. The optimal multi-stage information exchange strategy allows for a successful execution of the task of interest in 1722 days (4.7 years) on average with standard deviation of 385 days. The execution of this protocol took as few as 1004 and as many as 4965 with median of 1616 days.

Keywords: Large Scale Simulations, Classical Mathematics Puzzles, 100 prisoners and light bulb puzzle

Introduction

Conceptually, this research addresses a special case of the important problem of finding optimal design for dissemination of information among the subject of a population of interest under severe restrictive rules that include very limited amount of information allowed to be transferred at each step of the algorithm. In particular, we studied a novel version of the hundred prisoners and a light bulb classical mathematical puzzle, defined as follows:

One hundred prisoners have been newly ushered into prison. The warden tells them that starting tomorrow, each of them will be placed in an isolated cell, unable to communicate amongst each other. Each day, the warden will choose one of the prisoners uniformly at random with replacement and place him in a central interrogation room containing only a light bulb with a toggle switch. The prisoner will be able to observe the current state of the light bulb. If he wishes, he can toggle the light switch. He also has the option of announcing that he believes all prisoners have visited the interrogation room at some point in time. If this announcement is true, all prisoners are set free, but if it is false, all prisoners are executed. The warden leaves and the prisoners huddle together to discuss their fate.

Can they agree on a protocol that will guarantee their freedom in the shortest amount of time?

Clearly, the switch can display only two states (say 0 and 1) and at each distinct step of a successful scheme, only two different state transitions are possible, the single switch position being changed from 0 to 1 or vice versa. Extensive study of protocols and their properties has already been accomplished (Wu, 2002), (Wu, Wu riddles, 2002). Moreover, previous research work has proposed and discussed several variations and generalizations of this classical problem (Paul-Olivier Dehaye et al., 2003).

In this study, we investigate the same problem with the novel assumption that two light bulbs with two separate switches are present in the interrogation room which allows us to use combinations of switch positions to display four states (say 0, 1, 2, and 3 expressed by the binary representations 00, 01, 10 and 11 respectively) and thus, at each step of a successful protocol, twelve different state transitions are possible. The new two-light bulb setting allows for the design of several influential new approaches for strategy optimization that combined with the range of time points assigned to their actual implementation, result in a great variety of potential best protocols. We empirically estimated the distribution of the number of the days until the successful completion of each competing strategy through large-scale simulations due to their inherent analytically intractable complex behavior. All calculations were carried out using the R statistical package (R Development Core Team, 2011) (http://www.r-project.org).

Methods and Results

In the subsequent presentation we describe and analyze the performance of five protocols of increasing complexity and efficiency. The first two strategies have direct counterpart designs in the one light bulb setting which allows us to also assess the improvement in the time until their successful completion directly attributable to the additional information transferable via the second light. In the successive sections we use the term strategy, design, algorithm, scheme and protocol interchangeably. In all protocols, we conceptualize the problem setting as initially assigning a single token to each prisoner representing the fact that they have not been counted as subjects that have visited the interrogation room.

Prisoners can deposit their token by increasing the value of the switch in certain design rounds or by not interacting with the switch in other stages. The collection of the tokens is assigned to one or more counters that can manipulate the state of the switch in order to accumulate predetermined quotas or deposit these quotas for future collection by a master counter. Depending on the strategy scheme and stage, prisoners can decrease the value of the switch (consequently adopting extra tokens) in an effort to extend the continuity of a current effective algorithm step. When multiple courses of action are possible, we give priority to single tokens over quota depositions being displayed on the switch as usually more than one counter can collect the single token but only one master counter can retrieve the quota from the switch. Lastly, in multi-stage algorithms, the prisoner arriving on the first day of an adjacent stage, manipulates the switch and his own count of single tokens as a technique that permits a correct transition between stages during which the switch states represent different quantities.

1. Single Preselected Counter Protocol

This simple protocol preselects a leader who is responsible for collecting the tokens of the rest of the prisoners (drones). More formally, let SW denote the value represented by the combination of the two switches. Then,

a) If a drone enters and has a token and $SW<3$, he deposits his token by incrementing the SW by 1.
b)If the leader enters, he collects the number of tokens represented by the SW and adds it to his count. He then sets the SW to 0.
c)The leader declares victory when the number of collected tokens reaches 100.

This protocol does not depend on any parameters. We simulated 100,000 implementations of this strategy (summary statistics presented in Table 1). Our results show that using this design, the average number of days until the leader declares victory is 3722 (an improvement of 6696 days or 64% over the comparable protocol with one light) with standard deviation of 581. The execution of protocol 1 took as few as 1786 and as many as 6848 with median of 3690 days.

2. Single Dynamically Selected Counter Protocol

Similar to the one switch protocols, we can improve on the two-switch Single Counter Protocol by choosing to elect a leader during a special snowball strategy round rather than preselecting him. In contrast to the previous protocol where the switch counts new visitors, during the snowball round, the switch is incremented by 1 every time a repeated visitor enters the interrogation room. The leader is selected to be the first person to set the switch to 3. The efficiency benefit of this approach is underlined by the fact that during the first 71 days, the probability of selecting a new prisoner is greater than the probability of selecting a repeated visitor (based on simulation study not shown here). Thus, it is advantageous to count the less likely event of repeated visitors and not maximize the value on the switch quickly. Once the leader has been selected and the snowball round of predetermined length has finished executing, we continue the protocol with subsequent stage identical to protocol 1. More formally,

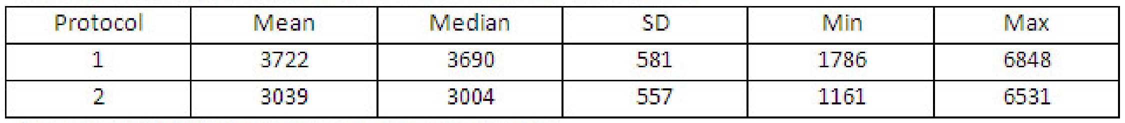
Stage 1 (snowball round) of length n days:
For days 1 through *n*-1, if SW<3:
If the prisoner who enters has a token, he leaves the SW as is, effectively depositing token.
If the prisoner who enters does not have a token, he increments the SW by 1. If this would make SW=3, this prisoner becomes the leader and initializes his count to *k*-3, where *k* is current day of the snowball round.
If SW=3, whoever enters does nothing.

For day *n*:
If SW < 3, whoever enters becomes the leader, setting his count at *n*-SW.
If SW = 3, whoever enters does nothing.
SW is now set to 0 for round 2.

Stage 2:
Begins at day *n*+1 and continues as the normal Single Counter Protocol does.

This protocol depends on a single parameter that denotes the length of the snowball round. We simulated 100,000 implementations of this strategy for snowball round lengths ranging from 20 to 40 days. This range is selected based on the distribution of the number of of days necessary for the switch to fill up. Based on a simulation study not presented here, we concluded that the duration of the snowball round that minimizes the time until a successful completion of the protocol is 35. Our results show that using optimal version of this design, the average number of days until the leader declares victory is 3039 (an improvement of 6264 days or 67% over the comparable protocol with one light and an improvement of 683 days or 18% over protocol 1) with standard deviation of 557. The execution of protocol 2 took as few as 1161 and as many as 6531 with median of 3004 days (summary statistics presented in Table 1).

Table 1. Summary statistics of the results for protocols 1 and 2 *.



Protocol	Mean	Median	SD	Min	Max
1	3722	3690	581	1786	6848
2	3039	3004	557	1161	6531

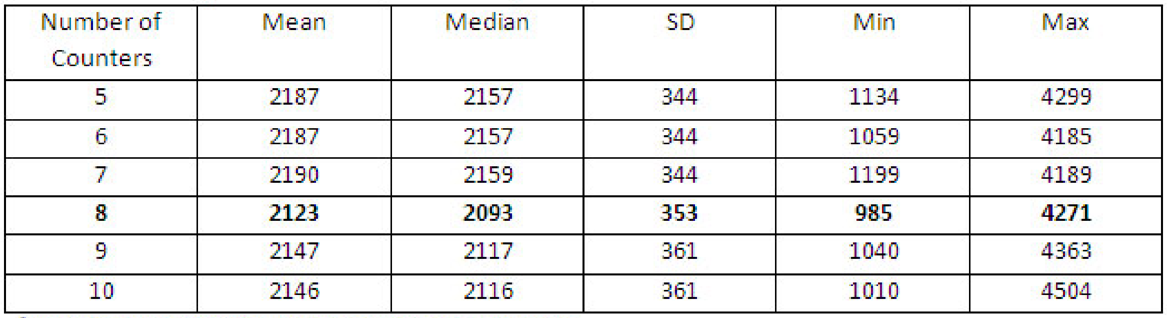
*Based on 100,000 simulation samples per scenario.

3. Multiple Preselected Counters Protocol

The inherent flaw of protocol 1 is that often the switch is loaded and awaiting the arrival of the leader who enters the interrogation room every 100 days on average and during that time the new visitors cannot deposit their tokens. As discussed in Wus paper 100 Prisoners and a Light Bulb, an improvement can be attained by preselecting *m* assistant counters (or ACs) who collect the single tokens until they reach predetermined quotas. Once they reach their quotas, they can deposit them for future collection by the leader. Therefore, we reserve one of the switch states to denote a single quota, i.e. in this protocol the assigned switch values are as follows, SW=(0 tokens, 1 token, 2 tokens, 1 quota). The leader declares victory as soon as he collects all *m* quotas from the ACs.

This protocol depends on a single parameter that denotes the number of ACs. We simulated 100,000 implementations of this strategy for 5, 6, 7, 8, 9, and 10 assistant counters. Our results show that the optimal version of this design is attained by 8 counters and the corresponding average number of days until the leader declares victory is 2123 (an improvement of 916 days or 30% over the fastest variant of protocol 2) with standard deviation of 353. The execution of protocol 3 took as few as 985 and as many as 4271 with median of 2093 days (summary statistics presented in Table 2).

Table 2. Summary statistics of the results for protocol 3*.



Number of Counters	Mean	Median	SD	Min	Max
5	2187	2157	344	1134	4299
6	2187	2157	344	1059	4185
7	2190	2159	344	1199	4189
8	**2123**	**2093**	**353**	**985**	**4271**
9	2147	2117	361	1040	4363
10	2146	2116	361	1010	4504

*Based on 100,000 simulation samples per scenario.

4. Multiple Dynamically Selected Counters Protocol

In essence, this protocol combines and exploits the advantageous properties of strategies 2 and 3. This scheme starts with *m* consecutive snowball rounds designed to dynamically elect *m* counters (the leader and the remaining *m*-1 assistant counters). After the end of all snowball rounds, the *m* counters have been selected and the protocol continues with stage 2 in a fashion identical to that of protocol 4. As mentioned earlier, based on an additional simulation study, during the first 71 days, the probability of selecting a new prisoner is greater than the probability of selecting a repeated visitor which shows that the successive snowball rounds should extend around 71 days in total. Further, we completed a separate simulation study (results not presented) to assess the average number of days necessary for the switch to consecutively fill up. Based on these two ancillary results, for 7, 6, 5 and 4 snowball rounds, we define the following snowball round lengths (26,11,9,8,7,7,6), (28,12,10,9,8,7), (30,14,11,10,9) and (33,16, 13,13) respectively. Lastly, we determine and assign the corresponding quotas to the counters by uniformly distributing the expected number of uncollected tokens which produces the following values (29,15,13,12,11,11,9), (30,17,15,14,12,12), (33,20,17,16,14) and (37,24,20,19).

This protocol depends on a several parameters, number of snowball rounds, snowball rounds lengths, and quota values. In this analysis, we simulated 100,000 implementations of this strategy for 4, 5, 6 and 7 snowball rounds and fixed the corresponding lengths and quotas to the values determined by the above-mentioned arguments. Our results show that the optimal version of this design is attained for 6 snowball rounds and the corresponding average number of days until the leader declares victory is 1871 (an improvement of 252 days or 12% over the fastest variant of protocol 3) with standard deviation of 374. The execution of protocol 4 took as few as 816 and as many as 4727 with median of 1871 days (summary statistics presented in Table 3).

Table 3. Summary statistics of the results for protocol 4*.

Number of Counters	Mean	Median	SD	Min	Max
4	1986	1935	407	899	4772
5	1899	1855	386	777	4483
6	**1871**	**1829**	**374**	**816**	**4727**
7	1884	1845	373	888	4256

*Based on 100,000 simulation samples per scenario.

5. Multiple Dynamically Selected Counters Combined with Three Additional Stages of Distinct Switch Allocation States Protocol

This protocol is a multi-parameter and multi-stage advancement of protocol 4 for which we optimize the intricate interplay of all strategy-defining variables. We begin the algorithm with a fixed number of snowball rounds identical to stage 1 of design 4. Next, we incorporate a stage 2, characterized with SW=(0 tokens, 1 token, 2 tokens, 3 tokens), which allows us to fully exploit the switch capacity during the period when no quotas have likely been reached. Further, we integrate a stage 3, characterized with SW=(0 tokens, 1 token, 2 tokens, 1 quota), which allows the deposit of the first available quotas while also permitting for the switch to collect the likely 1 or 2 single tokens for collection by the counters. Lastly, we incorporate a stage 4, characterized with SW=(0 tokens, 1 token, 1 quota, 2 quotas), which allows the deposit of the likely already accumulated 1 and 2 quotas while also permitting for the switch to collect the last remaining single tokens. Evidently, this protocol depends on the following set of parameters, number of snowball rounds (N), duration of all snowball rounds (S=(s_1, s_2, , s_N)), quotas assigned to each counter (Q=(q_1, q_2, , q_N)), duration of stage 2 (D2), and duration of stage 3 (D3). We fixed the quotas to the values determined via the argument presented in protocol 4 and created a grid of parameter values consisting of all possible 1024 combinations of the following quantities, 4, 5, 6 and 7 snowball rounds, 16 vectors of snowball lengths that cover sets of values centered about the quantities argued in protocol 4, 200, 400, 600 and 800 days for stage 2 lengths, 100, 300, 500, and 700 days for stage 3 lengths. We simulated 10,000 implementations for this strategy for each of the parameter combination defined above (summary statistics presented in Table 4). Interestingly, the top 25 best performing protocol variants were derivations of the 6 snowball round strategy. Our results show that the optimal design is given by N=6, S=(26,10,8,7,6,5), Q=(31,17,15,14,12,11), D2=800, D3=300. This best performing version of scheme 5 attained the lowest average number of days until the leader declared victory of 1706 (an improvement of 165 days or 9% over the fastest variant of protocol 4) with standard deviation of 385. The execution of protocol 5 took as few as 1004 and as many as 4965 with median of 1616 days.

Table 4. Summary statistics of the results for the top 3 (out of the analyzed 1024) best performing versions of protocol 5*.

Combination of number of counters (N), lengths of snowball periods (S), quotas (Q), length of stage 2 (D2), length of stage 3 (D3)	Mean	Median	SD	Min	Max
N=6, S=(26,10,8,7,6,5), D2=800, D3=300	1706	1616	385	1004	4965
N=6, S=(26,10,9,8,7,7), D2=800, D3=300	1710	1617	387	1041	5308
N=6, S=(25,10,8,8,7,7), D2=800, D3=300	1711	1617	392	1062	4453

*Based on 10,000 simulation samples per scenario.

Discussion

We have performed a large scale simulation study to investigate the performance of thousands of parameter-based variations of 5 competing protocols that successively incorporate various advantageous techniques and attain progressively increasing design sophistication and performance. The average times required for these strategies to complete the task of interest were 3722, 3039, 2123, 1871 and 1706 days with corresponding consecutive improvements of 18, 30, 12, and 9% respectively. However, we realize that the optimal strategy that emerged from our analysis is not the most efficient one. In the subsequent discussion we outline several algorithms and design steps that will potentially decrease the number of days until victory is declared.

We can instruct a dynamically selected counter that finds the SW nonempty at a later snowball round, to reduce both his token count and the SW value by the same number in an effort to permit the current efficient collection of tokens to continue. In particular, if the switch is to become full (which would force him to be elected as a counter again), the subtraction of tokens from his count obtained in a previous snowball round would allow the current snowball round and associated search for a counter to continue. Further, if a dynamically selected counter is to become elected for a second time on the last day of another snowball period and SW<3, he can increment the SW by one to inform the prisoner at the start of the next snowball (or the first prisoner of Stage 2 if this is the last snowball phase) that they are responsible for the quota and the count represented by the SW. This process can be extended 3 days into the next round (by assigning SW values exceeding the values possible under the assumption of a normal counter selection in the previous snowball round) in an effort to reduce the probability of reselecting counters (which dramatically increases the completion time of all protocols).

Although it seems obvious that a counter should implement the first strategy outlined above, it is not clear during which part of the snowball round this approach is really beneficial or exactly how many tokens he should reduce his count and the SW by. For instance, the adoption of this approach seems insightful if a previously elected counter enters very early on in a snowball round and finds SW= 2 as there is a great chance that the switch will become full long before the end of the snowball round, thus resulting in many unproductive days for the remainder of the round during which no one can do anything. However, if it is in the last couple of days of a snowball round when this counter enters and SW=1, there is much less danger of the SW becoming full, and even if it does, there are very few unproductive days left. Thus, there appears to be little upside for the implementation of the reduction strategy in this case. In fact, it is entirely possible that there is even a downside for such decision, as it would likely detrimentally affect the efficient execution of stage 2 which depends on the presence of approximately identical differences between all counters counts and quotas. If these differences are imbalanced, some counters will meet their quotas too early and therefore relinquish themselves of the capability to collect tokens and free up the switches for uninterrupted, continuous use. This means that a counter should be conscious and make decisions based on difference of his quota and count. He would only have true information about his own count and would have to use the expected counts for other snowball rounds given his observed information. Thus, he should manipulate the SW to assure to the best of his knowledge that all counters have similar differences between their quotas and counts. The number of tokens a returning counter should deposit in a snowball round when SW<3 is another parameter that should be varied and examined through simulations in order to best obtain a uniformly optimal protocol.

If a counter elected in an earlier snowball round comes in to find the SW full, he can set the SW to 0, 1, or 2, effectively depositing k SW of his tokens where k is the number of days since the start of the current snowball round. For the remainder of the snowball round, everyone who enters will have assumed that no counter has been elected for the round and act accordingly. This essentially allows a counter elected from a previous snowball to transfer his responsibility to someone else in a later snowball round all the while not interfering with that round standard counter selection. However, because everyone for the remainder of the snowball will have no way of differentiating this case from normality, the counter elected from the previous snowball can only pass his responsibility to this round if his quota and the quota of the current snowball are the same. Also, since this counter is essentially relinquishing himself of all his responsibilities to his quota, he must make sure that k SW is less than or equal to his count. Otherwise, he will be left with a negative count and no quota and consequently no means to convey to everyone else that he has a negative count which might produce a false positive victory. Further, if k SW is less than his count, it might not necessarily be advantageous for him to pass his quota depending on how many remaining tokens he is left holding as he will have to enter the room a number of times just to deposit his remaining tokens. If k SW is equal to his count, then it is definitely worth for him to pass the quota, because doing this allows the collection of more tokens in the current snowball round to continue with no down side. Lastly, the leader cannot pass his quota, because the person receiving the quota will have no idea that he is also being transferred the leaders obligation.

There is also another snowball technique that is not available to the one switch schemes. It is safe to say that near the end of a single snowball round, it is likely that SW>0. This means that at a certain point, it would be more practical to reallocate the state of 0 from meaning that 0 repeated visitors have entered the interrogation room to meaning that 3 repeated visitors have done so. If however, we arrive at this point and SW=0 (representing 0 repeats), whoever enters can simply pretend he does not have a token and increment the SW by 1 so that during the next phase of the snowball round, the prisoners do not think that the SW is representing 3 repeated visitors. This concept of reallocating permutations of the SW states during a snowball round can actually be done multiple times. This technique will allow us to extend the duration of each individual snowball round before the SW will become full. The optimal positions of the reallocation points of the SW states needs to be studied in future endeavors.

As discusses earlier, the traditional snowballs are no longer practical when extended beyond day 73, as the probability of selecting new prisoners becomes smaller than that of choosing repeated visitors. Thus, we can prolong the assistant counters selection in reverse snowball rounds during which we count the new visitors instead of repeated ones in a fashion similar to stage 2 of protocol 5, i.e. SW=(0 tokens, 1 token, 2 tokens, 3 tokens). Further, we can split the round into several phases comparable to the technique used in traditional snowball rounds with reallocation points. The end of each phase of a reverse snowball round should be assigned on a day by which the SW becomes full with high probability. If the SW is full by the end of a given phase, then the SW is reset to 0 and all tokens conveyed by the SW are deposited for collection by the counter. This means that everyone in the next phase would know that there are 3 tokens in the bank as well as what the SW is showing on the day they are summoned to the interrogation room. This design allows for a gradual accumulation of tokens eventually collected by the AC at the end of the entire round. However, if the SW is not full and not empty at the end of a phase, the prisoner entering on the last day of the phase can decide to relay the current SW state to the next phase in an effort to borrow tokens form the future. The prisoner that is forced to become the counter due to appropriate SW and time combinations, adjusts his count to match the sum of the current SW value and the total number of deposited tokens from the successfully completed previous phases and sets the SW to a fail setting (say 3) to convey to the prisoners entering in future phases that the AC for the round has already been elected.

Also, as mentioned earlier, previously elected ACs from traditional or reverse snowball rounds can manipulate the switches by depositing tokens of their own to help prolong the productivity of the currently executing round. Lastly, it is obvious that at some point these types of snowballs will become impractical and we will need to set the starting point for Stage 2 where ACs and the leader accumulate and deposit their quotas.

As evidenced by the methods, results and discussion presented, the two-light setting of the hundred prisoners problem includes multifarious design options that encompasses a large number of protocol (and corresponding varieties) inducing parameter combinations. Large-scale future studies would enable us to advance our understanding of the complex interdependences among the strategy defining parameters described above.

References

Wu, W. (2002). 100 Prosoner and a light bulb. *Unpublushed*. http://www.ocf.berkeley.edu/~wwu.

Wu, W. (2002). Retrieved from Wu riddles: http://www.wuriddles.com.

Paul-Olivier Dehaye, D. F. (2003). One hundred prisoners and a light bulb. *Mathematical Intelligencer, 24*(4), 53-61.

R Development Core Team. (2011). *R: A Language and Environment for Statistical Computing.* R Foundations for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **Bui**

The Bernoulli Family: their massive contributions to mathematics and hostility toward each other

Dung (Yom) Bui, Mohamed Allali

Abstract

Throughout the history of mathematics, there are several individuals with significant contributions. However, if we look at the contribution of a single family in this field, the Bernoulli probably outshines others in terms of both the number of mathematicians it produced and their influence on the development of mathematics. The most outstanding three Bernoulli mathematicians are Jacob I Bernoulli (1654-1705), Johann I Bernoulli (1667-1748), and Daniel Bernoulli (1700-1782), all three of whom were the most influential math experts in the academic community yet very hostile to each other. Their family structure and jealousy toward each other might have fueled their massive contributions to mathematics.

Keywords: Bernoulli family, history of mathematics, Jacob Bernoulli, Johann Bernoulli, Daniel Bernoulli

Introduction

The Bernoulli family was originally from Holland with strong Calvinism religion. In 1567, Philip, the King of Spain, sent a large army to punish those who were opposed to Spanish rules, enforced adherence to Roman Catholicism, and re-established his authority. To avoid Spanish religious persecution, the Bernoulli fled to Basel in Switzerland, which at that time was a great commercial hub of central Europe (1-2).

Initially, members of the Bernoulli family were successful traders and prospered in business. All of the Bernoulli mathematicians were descendants of Nicholas Bernoulli, a prominent merchant in the spice business in Basel. Despite the non-mathematic family traditions, later members of the Bernoulli turned out to be the most influential math experts in the academic community (5).

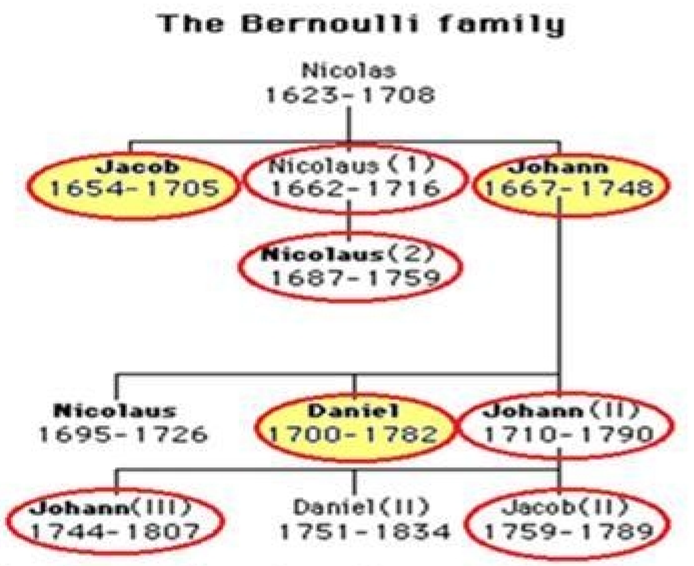


Fig 1. The Bernoulli Family Tree (the ones circled in red are mathematician members)

Within three successive generations, this family produced as many as eight mathematicians who contributed vastly to the mathematic world. Together with Isaac Newton, Gottfried Leibniz, Leonhard Euler, and Joseph Lagrange, the Bernoulli family dominated mathematics and physics in the 17th and 18th centuries, making critical contributions to differential calculus, geometry, mechanics, ballistics, thermodynamics, hydrodynamics, optics, elasticity, magnetism, astronomy, and probability theory. Unfortunately, the elder Bernoullis were as conceited and arrogant as they were brilliant. They engaged in bitter rivalries with one another (7).

Among the eight Bernoulli mathematicians, the most famous and outstanding three were Jacob I Bernoulli (1654-1705), Johann I Bernoulli (1667-1748), and Daniel Bernoulli (1700-1782). In this paper, we will focus our discussion on these three mathematicians (5-8).

Jacob Bernoulli (1654-1705)

Jacob Bernoulli (1654-1705) - Nicholas Bernoulli's oldest son- was the first in the family to be recognized as a strong influential mathematician although there was no family tradition before him. His parents compelled him to study philosophy and theology, which he greatly resented but he did acquire a theology degree in 1676. After taking his theology degree, Bernoulli moved to Geneva where he worked as a tutor. He then travelled to France, spending two years studying with the followers of Descartes. In 1681 Bernoulli travelled to the Netherlands and then to England where, he met several mathematicians, including the famous scientists Robert Hooke (1635-1703) and Robert Boyle (1627-1691). As a result of his travels, Bernoulli developed correspondence with many mathematicians which he carried on over many years. He was intrigued with mathematics and astronomy so much he grew interest in these fields regardless of his father's opposition. Jacob Bernoulli returned to Switzerland and taught mechanics at the University of Basel since 1683. Although his degree was in theology and he was offered an appointment in the Church, he turned it down. Bernoulli's real love was for mathematics and theoretical physics and it was in these topics that he taught and researched (2-5)

Jacob Bernoulli published five treatises on infinite series between 1682 and 1704. The first two of these contained many results, such as fundamental result that S(1/n) diverges, which Bernoulli believed was new but they had actually been proved by Mengoli 40 years earlier.

In May 1690, in a paper published in Acta Eruditorum (the oldest scientific journal that began appearing since 1682 under the supervision of Gottfried Leibniz (1646-1716)), Jacob Bernoulli showed that the problem of determining the isochrone is equivalent to solving a first-order nonlinear differential equation. The isochrone, also called curve of constant descent, is the curve along which a particle will descend under gravity from any point to the bottom in exactly the same time no matter where it starts on the curve. It had been studied by Huygens in 1687 and Leibniz in 1689. Also in this paper, Jacob Bernoulli was the first mathematician to use the term "integral calculus". Earlier, Leibniz had designated it by "Calculus summatorium" (8).

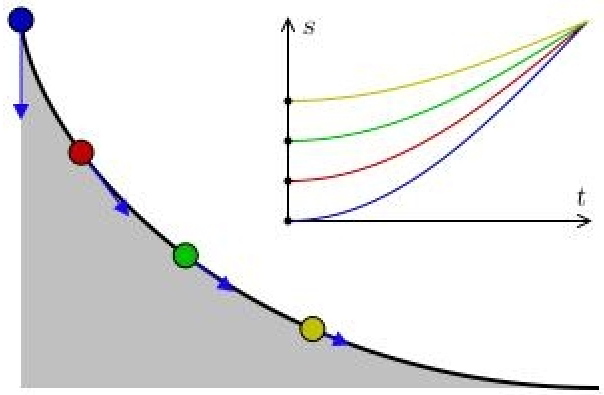


Fig 2. The isochrone

After finding the differential equation, Bernoulli then solved it by what we now call separation of variables. Jacob Bernoulli's paper of 1690 is important for the history of calculus, since that is where the term "integral" appears for the first time with its integration meaning. In 1696, Jacob Bernoulli solved the equation now called "the Bernoulli equation":

$$y' = p(x)y + q(x)y^n$$

Jacob used differential equations in tackling various geometrical and mechanical problems (4).

The Catenary problem was also one of the problems which generated a lot of enthusiasm among mathematicians after the invention of calculus. The word "catenary" comes from the Latin word "catena," which means chain. In the May 1690 issue of "Acta Eruditorum", Jacob wrote "And now let this problem be proposed: To find the curve assumed by a loose string hung freely from two fixed points". Earlier, Galileo (1564-1642) had dealt with this problem and wrongly concluded that the curve would be a parabola. In 1646, Christian Huygens (1629-1695), at the age of sixteen, proved that the curve cannot be a parabola. One year after the appearance of the problem, in the June 1691 issue of "Acta Eruditorum," three correct solutions were published. The three solutions came from Huygens, Leibniz, and Johannn Bernoulli (Jacob's younger brother). Although they approached the problem from three different points of view, all of them concluded that the curve would be a catenary. Jacob himself failed to solve the problem. The Cartesian equation of a catenary is:

$$y = \frac{e^{ax} + e^{-ax}}{2a}$$

Where, *a* is a constant whose value depends on the mass per unit length of the chain and the tension of suspension (9).

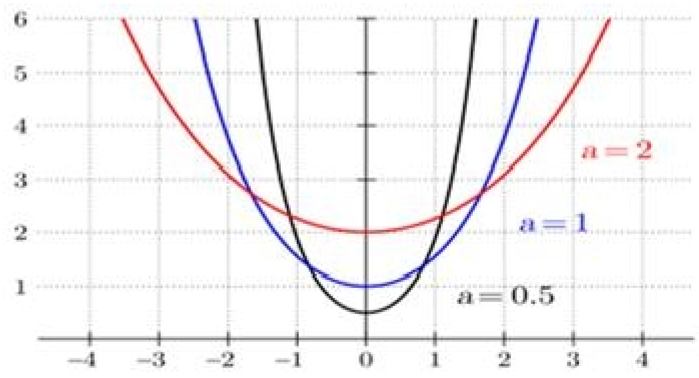


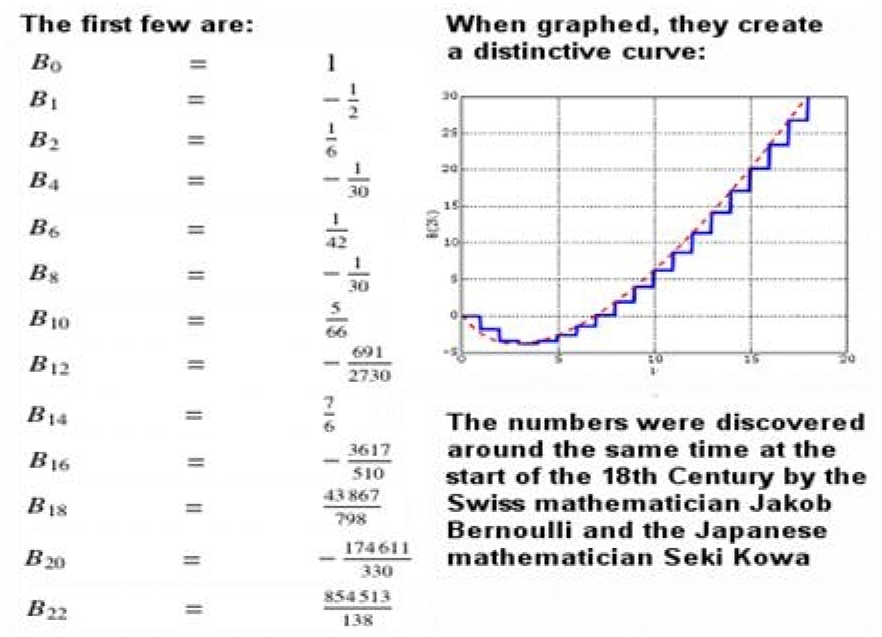
Fig 3. The catenary

Fig 4. Examples of the catenary in real life

The study of the catenary problem was soon applied in the building and designing of suspension bridges.

Jacob Bernoulli was one of the pioneers of the mathematical theory of probability. His first paper on probability theory was published in 1685. The greatest contribution of Jacob Bernoulli is *Ars Conjectandi* (published posthumously, 6 years after his death in 1713; also called "The Art of Conjecturing") contained many of his finest concepts: his theory of permutations and combinations; the so-called Bernoulli numbers, by which he derived the exponential series; his treatment of mathematical and moral predictability; and the subject of probabilitycontaining what is now called the Bernoulli law of large numbers, basic to all modern sampling theory (6-8).

Bernoulli numbers: $$B_m(n) = \sum_{k=0}^{m} \sum_{v=0}^{k} (-1)^v \binom{k}{v} \frac{(n+v)^m}{k+1},$$



The first few are:

B_0	=	1
B_1	=	$-\frac{1}{2}$
B_2	=	$\frac{1}{6}$
B_4	=	$-\frac{1}{30}$
B_6	=	$\frac{1}{42}$
B_8	=	$-\frac{1}{30}$
B_{10}	=	$\frac{5}{66}$
B_{12}	=	$-\frac{691}{2730}$
B_{14}	=	$\frac{7}{6}$
B_{16}	=	$-\frac{3617}{510}$
B_{18}	=	$\frac{43867}{798}$
B_{20}	=	$-\frac{174611}{330}$
B_{22}	=	$\frac{854513}{138}$

When graphed, they create a distinctive curve:

The numbers were discovered around the same time at the start of the 18th Century by the Swiss mathematician Jakob Bernoulli and the Japanese mathematician Seki Kowa

Fig 5. The Bernoulli numbers are a sequence of rational numbers with deep connections to number theory.

In 1689, Jacob Bernoulli published many research papers on the theory of infinite series. He was the first to think about the convergence of an infinite series and proved that the series

$$\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots$$

is convergent. He was also the first to propose continuously compounded interest, which led him to investigate:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n.$$

Using the binomial expansion of

$$\left(1+\frac{1}{n}\right)^n,$$

he showed that this limit lies between 2 and 3, although the value of the limit would need to wait until Euler and it is equal to e.

Jacob also showed special interest in the logarithmic spiral, which was introduced in 1637 by Rene Descartes (1596-1650). At that time, the polar equation of the curve was written as

$$r = e^{a\theta}$$

Where *q* is measured in radians and *a* denotes the rate of increase of the spiral.

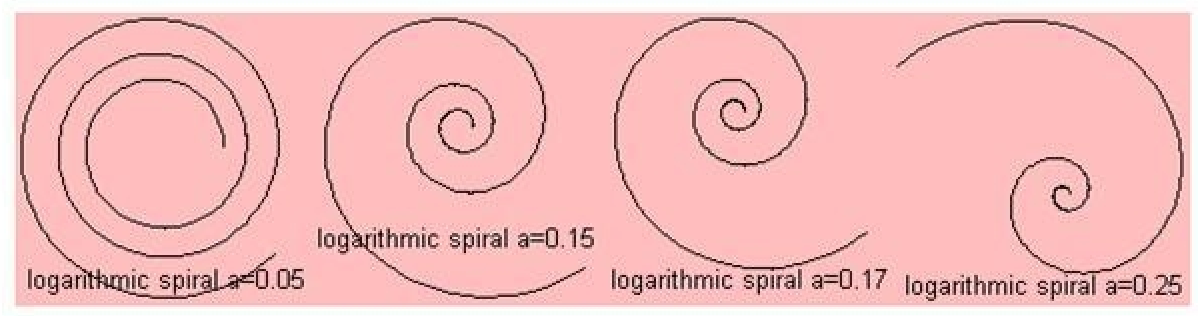

logarithmic spiral a=0.05
logarithmic spiral a=0.15
logarithmic spiral a=0.17
logarithmic spiral a=0.25

Fig 6. The logarithmic spiral

The property of the logarithmic spiral which surprised Jacob Bernoulli the most is that it remained unaltered under many geometrical transformations. For example, typically a curve suffers a drastic change under inversion but a logarithmic spiral generates another logarithmic spiral under inversion, which is a mirror image of the original spiral. The evolute, pedal, and caustic of a logarithmic spiral are the very same spiral. Jacob Bernoulli was so astonished by the "magic" of the logarithmic spiral that he requested to have a figure of a logarithmic spiral engraved on his tombstone with the words *Eadem mutata resurgo ("Changed and yet the same, I rise again")* written under it. He wrote that the logarithmic spiral 'may be used as a symbol, either of fortitude and constancy in adversity, or of the human body, which after all its changes, even after death, will be restored to its exact and perfect self." However, unfortunately, possibly from ignorance, the artisan engraved an Archimedian spiral instead (7-8).

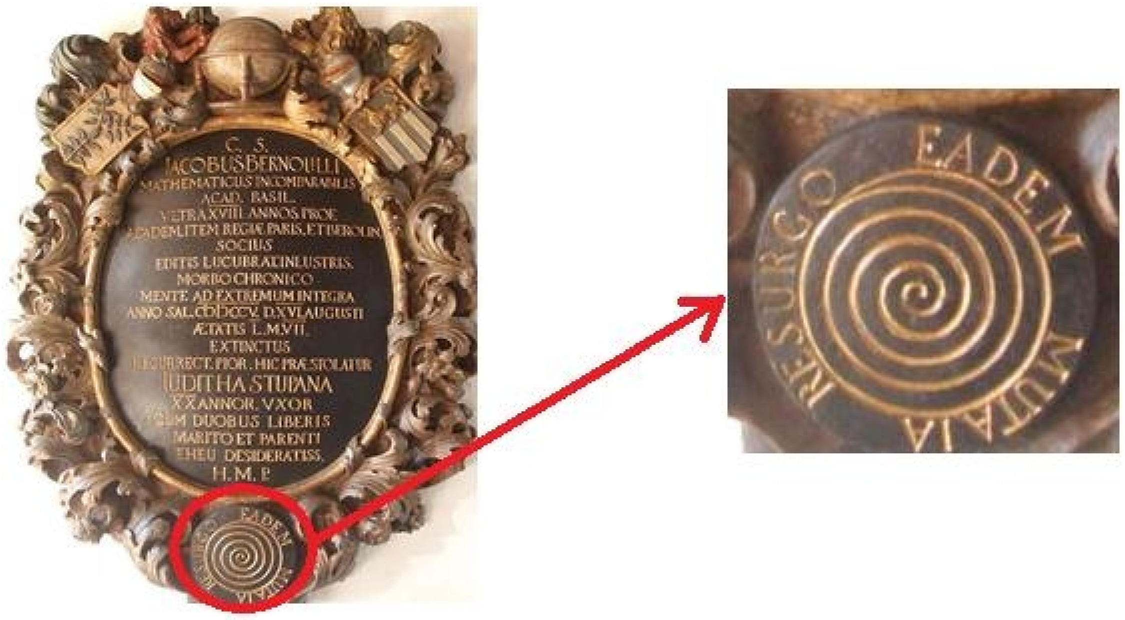


Fig 7. Jacob Bernoulli's gravestone

Jacob Bernoulli became the chair of mathematics at Basel in 1695 and held the position until his death in 1705 when the position was taken over by his younger brother, Johann I Bernoulli.

Johann I Bernoulli (1667-1748):

Johann Bernoulli was Jacob Bernoulli's younger brother. He was a brilliant mathematician who made important discoveries in the field of calculus. His father, Nicholas, again provided Johann with a well-rounded education so that he could eventually enter the family business. However, Johann had no desire to work in the spice industry, and enrolled in the University of Basel in 1683 to study medicine.

He asked his brother, Jacob, to teach him mathematics. By this time, Jacob, who was twelve years older than Johann, was a professor of mathematics at the University of Basel. With Jacob's help, Johann studied Leibniz's papers on calculus until he became proficient, and his mathematical prowess soon matched that of his brother's (2-5*)*.

At the time, calculus was a difficult subject to understand. The Bernoulli brothers were among the few mathematicians in Europe to understand calculus. At first, Jacob had no problem teaching his little brother. He realized his brother's talents and quick-study of mathematics so he offered to work with Johann. The cooperation between the two brothers soon degenerated, however, into vitriolic arguments. As time went on, Johann's ego was getting larger. He began to brag about his work and at the same time belittled his brother. Jacob was so angry he made crude comments about Johann's abilities, referring to him as a student repeating what the teacher taught him, in other words, a parrot. Jacob and Johann went back and forth with insulting comments about each other in the academic community which developed a notorious reputation of their family togetherness (2-7).

Despite family problems, Johann was an excellent mathematician. Johann was perhaps even more productive as a scientist than was Jacob. He studied the theory of differential equations, discovered fundamental principles of mechanics, and the laws of optics. He used calculus to solve problems which Newton failed to solve regarding the laws of gravitation. Johann also did lots of important work on the study of the equation

$$y = x^x.$$

He discovered the Bernoulli series and made advances in theory of navigation and ship sailing. In addition, he is famous for his tremendous letter writing (over 2500 messages) and his tutoring of another great mathematician, Leonhard Euler (9).

Johann proposed the so-called brachistrochrone problem, the curve of most rapid descent of a particle sliding from one point to another under the influence of gravity. Johann invited the "shrewdest mathematicians all over the world" to solve this problem and allowed six months time for it. Five correct solutions were received. Of these, four came from renowned mathematicians, L'Hôpital, Leibniz, Jacob and Johann himself and one from an anonymous mathematician. However, seeing the style of tackling the problem, experts could recognize that the anonymous solver was no one but Sir Isaac Newton and Johann commented on it "Ex ungue Leonem" (Tell a lion by its claw). Newton later admitted that he solved the problem by thinking relentlessly for 12 hours over it. The answer to this problem is a fragment of an inverted cycloid. A cycloid is the curve traced by a fixed point on the perimeter of a wheel rolling along a straight line (2-3).

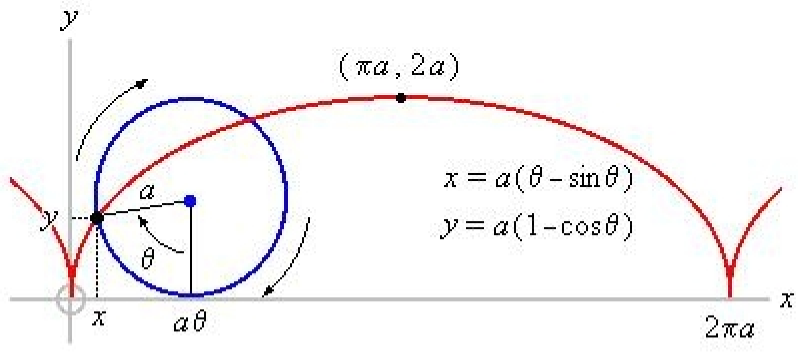


Fig 8. A cycloid is the locus of a point P on a circle as the circle rolls along a line

Johann's method of solving the brachistochrone problem was revolutionary in that it formed the basis for a new branch of calculus, calculus of variations. Calculus of variations, a generalization of calculus, deals with extremizing functionals, as opposed to ordinary calculus which deals with functions. In 1673, twenty-three years after the brachistotone problem, Huygens discovered that the cycloid is also the solution of the "tautochrone problem". In the "tautochrone problem," it is required to find a curve so that a particle, moving under gravity, will reach a given point on the curve in equal time irrespective of its initial position on the curve.

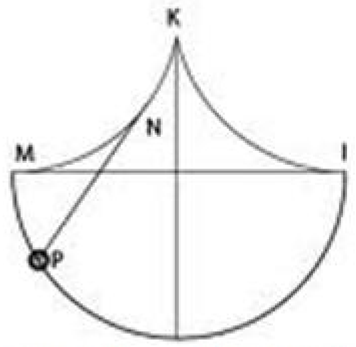


Fig 9. The tautochrone curve

Knowing that the cycloid is the solution of both the "brachistotone problem" and the "tautochrone problem," Johann was moved to write: "But you will be petrified with astonishment when I say that exactly this same cycloid, the tautochrone of Huygens, is the brachistotone we are seeking" (2).

In 1691, Johann was introduced to Marquis de L'Hôpital, a young mathematician with a keen interest in calculus, who asked Johann if he would give him lectures on the new integral calculus and the differential calculus of which he had heard very little about. L'Hôpital had heard little about the calculus due to the war going on in Europe at the time, which slowed the transmission of information dramatically.

While tutoring L'Hôpital, Johann signed an arrangement saying that he would send all of his discoveries to L'Hôpital to do what he wished, in return for a regular salary. This arrangement resulted in one of Johann's biggest contributions to the calculus being known as L'Hôpital's rule. Hence, although Johann "discovered" this relationship, it was L'Hôpital who got the credit for it (*7).*

Johann Bernoulli also used his position of authority to act as an advocate for Leibniz, who was embroiled in a battle with Isaac Newton over ownership of the credit for inventing calculus. There was a debate in mathematical circles about who the true inventor was, and whether Leibniz plagiarized Newton's work, and vice-versa. Bernoulli publicly criticized Newton's calculus in scientific journals, and demonstrated that Leibniz's version of calculus, which used differential notation, was superior by using it to solve problems that Newton could not. Bernoulli's influence was so great that Leibniz's methods became the first choice in Europe (5-6).

Johann wanted the chair of mathematics at University of Basel which was held by his brother but he was unable to get it so Johann accepted a chair at the University of Groningen in Netherland. He vowed not to come back to Basel but in 1705, Johann's father-in-law was dying and asking for his daughter and grandchildren, so Johann came back to Basel. While traveling, he did not know his brother, Jacob, died of tuberculosis. Once he realized his brother's death, Johann took over his position. During his life, Johann was awarded many honors including membership at the Academies of Science at Paris, Berlin, St. Petersburg, and many others. He passed away in Basel on the first day of the year 1748 (8-9).

Daniel Bernoulli (1700-1782):

Daniel Bernoulli was Johann Bernoulli's second son. He was born in Groningen, Netherlands, on February 8, 1700. As his own father did to him, Johann Bernoulli did not want his son to enter the field of mathematics. Johann Bernoulli tried to convince Daniel that there is no money in mathematics and tried to place Daniel in an apprenticeship with a merchant, but Daniel resisted. Daniel began studying logic and philosophy at the University of Basel when he was thirteen and received his bachelor's degree two years later. In the following year, he obtained his master's degree. His father eventually allowed Daniel to study medicine, and for the next few years, Daniel studied in Heidelberg, Germany, and Strasbourg, France (1-4).

In the early 1720s, Bernoulli traveled to Venice to practice medicine. He also hoped to further his studies at the University of Padua, but he became too ill to follow this plan. Instead, he worked on mathematics, producing his first publication, "Mathematical Exercises," in 1724. Within a year, Bernoulli had also designed an hourglass for use at sea. The trickle of sand had to be constant, no matter how violently the ship might move, even in a storm. At the age of twenty-five, he submitted this work to the annual contest of the Paris Academy and won the grand prize.

As a result of this prize and the fame resulting from "Mathematical Exercises," Bernoulli was invited by Catherine I of Russia, widow of Peter the Great, to accept the chair of mathematics at the Imperial Academy in St. Petersburg. His brother, Nicolaus, was also offered a chair in mathematics at St. Petersburg at the same time (6).

Here, the brothers began to consider a game in which a coin is tossed "n" times until the first "heads" turns up. If a head occurs for the first time on the *n*th toss then you will be paid 2^n dollars. How much would you be willing to pay to play this game? Although the expected value of the payoff is infinite as:

$$\begin{aligned} E &= \frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 4 + \frac{1}{8}\cdot 8 + \cdots \\ &= 1 + 1 + 1 + \cdots \\ &= \infty . \end{aligned}$$

Of course, no one seems to be willing to pay this sum to play this game. So, Daniel Bernoulli's resolution of the paradox was to suppose that utility is not linear in the payoff but instead, strictly concave, e.g., ln(*x*). What certain payoff, *c*, would an individual with such a utility function regard as equivalent to the St. Petersburg lottery? This *c* would have to satisfy:

$$\begin{aligned}\ln(c) &= \sum_{i=1}^{\infty} (1/2)^i \ln(2^i) \\ &= \left(\sum_{i=1}^{\infty} (i/2^i)\right)\ln(2) \\ &= 2\ln(2) \\ &= \ln(4)\end{aligned}$$

And thus, c = 4 a rather modest value to place upon the infinite expectation. This is became known as the St. Petersburg paradox.

Tragically, Nicolaus died within a few months of the brothers' arrival in St. Petersburg so Johann Bernoulli arranged for his best student and also Daniel Bernoulli's childhood friend, Leonhard Euler to come help him. The next few years would be the most productive period of Bernoulli's life. While studying vibrating systems, he defined the simple nodes and frequencies of a system's oscillation, and demonstrated that the strings of musical instruments move in an infinite number of harmonic vibrations all along the string (3-7).

Between 1725 and 1749 Daniel won 10 prizes from the Paris Academy of Sciences for work on astronomy, gravity, tides, magnetism, ocean currents, and the behavior of ships at sea. He also made substantial contributions in probability.

The relationship between Johann and Daniel became strained when, in 1734, they both competed for the Grand Prize of the Paris Academy, a school of mathematical and physical sciences. They both won, and the thought of sharing the prize with his own son sent Johann into a rage. He banned Daniel from his house.

The relationship was damaged further when Johann Bernoulli published a piece on hydrodynamics in 1739 entitled, "Hydraulica" whereas one year earlier, Daniel published "Hydrodynamica," a very similar work. Johann blatantly plagiarized "Hydrodynamica" in his publication of "Hydraulica," and even made it seem as if his work was published first by backdating it to 1732. The plagiarism, however, was soon uncovered.

In his later years, Daniel Bernoulli continued to work in calculus, mechanics, conservation of energy, oscillation, and the movement of bodies in a resisting medium. He was a member of the London Royal Society; the Paris Academy of Sciences, the Berlin Academy, and the St. Petersburg Academy of Sciences. Bernoulli died in Basel on March 17, 1782, just a few weeks before his eighty-second birthday (8-9).

Conclusion:

The Bernoulli's competitive and combative family relationship is usually said to curtail their collaboration and hold back potential success which could have happened had they not been so ruthless to each other. However, perhaps their resentment and enviousness also challenged and inspired each other, resulting in their outstanding mathematic work. Had Jacob not been so excellent in math and at the same time, so envious of his brother, Johann would not have the same pressure and motivation to thrive and surpass Jacob and vice versa. The same argument also applies for Johann and his son, Daniel. Despite their jealousy and hostility toward each other, the Bernoulli family is undoubtedly the best mathematician family in terms of the number of mathematical geniuses it produced and the significant impact of their contributions to the academic community.

References:

1. Abbott, David Ph.D. ed. "Bernoulli, Jacques & Bernoulli, Jean." The Biographical Dictionary of Sciences: Mathematics. New York: Peter Bedrick Books, 1985. p. 17.

2. Mukhopadhyay, Utpal. "Jacob I and Johann I: A pair of giant mathematicians." Bernoulli Brothers. 7.9 (2001): 29-37.
3. Graham, Daniel. "Daniel Bernoulli and the St. Petersburg Paradox." St. Petersburg Paradox. 18.9 (2005): 18-27.
4. Sensenbaugh, Roger. "The Bernoulli Family." Great Lives from History Renaissance to 1900 Series. Englewood Cliffs, NJ: Salem Press, 1989. vol. 1, pp. 185-188.
5. "The Bernoulli Family." *http://www.math.wichita.edu/history/men/bernoulli.html*. Ed. Tina Gonzales. N.p., - . Web. 26 Jan. 2011.
6. "Bernoulli Brothers." *http://www.ias.ac.in/resonance/Oct2001/pdf/Oct2001p29-37.pdf*. Ed. Utpal Mukhopadhyay. N.p., - Oct. 2001. Web. 26 Jan. 2011.
7. "Bernoulli, Johan." *http://pirate.shu.edu/~wachsmut/ira/history/bernoull.html*. Ed. Paul Golba. N.p., 26 Mar. 2007. Web. 26 Jan. 2011.
8. "The Mathematical Bernoulli Family." *http://www.unisanet.unisa.edu.au/07305/fhome.htm*. Ed. Sharlene Faulkner. University of South Australia, - 1996. Web. 26 Jan. 2011.
9. "The Bernoulli Era." *http://noahbprince.com/aufiles/S09/S09HMNotesWk11.pdf*. N.p., - 2007. Web. 26 Jan. 2011.

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **Kristedja**

Utilizing Remote Sensing to Detect Deepwater Horizon Oil Spill Properties

Amanda Kristedja

Abstract

This paper focuses on the detection of oil spills using satellite information. For this research project, the focus will be primarily on the Deepwater Horizon Oil Spill, which is the largest accidental oil spill. In order to detect an oil spill, the chlorophyll *a* content must be observed with data from the SeaWiFs at 9km and the MODIS-aqua at 9km and 4km. There is conflicting evidence of whether or not chlorophyll a concentration is positively correlated with the presence of an oil spill. This will be further investigated in the experiment. Also, by utilizing the MODIS instrument aboard the terra and aqua satellites, the amount of solar radiation during the day can be measured, particularly at the wavelength 443nm. Also related to the reflectance of the area of the oil spill is the sea surface temperature. The sea surface temperature can be measured by the MODIS-aqua and MODIS-terra at 9km during the day and night. It is hypothesized that high levels of dissolved organic matter also suggest the presence of an oil spill. Measurements from the SeaWiFs at 9km and the MODIS-aqua at 9km and 4km can be analyzed to show the levels of dissolved organic matter. The measurements of these specific parameters help to detect oil spills as well as to show the effects that oil spills have on the surrounding environment. For this study, measurements from these satellites over the Gulf of Mexico are taken before, during, and after the occurrence of the Deepwater Horizon Oil Spill that occurred between April and July 2010. These measurements show that there are positive correlations between the presence of an oil spill and both the chlorophyll a concentration and reflectance of the oil spill. In contrast, sea surface temperature was found to be lower in the area of the oil spill. Evidence from colored dissolved organic matter in the affected area proved to be inconclusive.

Keywords: Oil Spill, Deepwater Horizon Oil Spill, MODIS-aqua, MODIS-terra, Chlorophyll

1. Introduction

Oil spills are a hazard introduced to the Earth by human activity. Oil spills are a pollutant specifically called liquid petroleum hydrocarbon. Sources of oil spills include oil platforms and rigs in the middle of the ocean as well as leaking from a crude tanker and other large ships. Although there are natural occurring oil seeps, these are usually over land rather than ocean and are not as much of a concern as the spills that are a result of human activity. Some environmental effects include the injuries and deaths of marine life. Also, if the oil sits on top of the water, it can hinder the sun's rays from reaching plant life and phytoplankton that reside on the Earth's ocean floor. Oil-consuming bacteria exist in nature and help in the process of removing the oil from the area. However, this is a slow process that merely acts as a cleaning agent, but does not prevent any of the harm to the marine life. Human efforts to clean these oil spills are present as well. There are many methods of extracting the oil from the ocean, yet some of them require specific conditions and others actually contribute to pollution.

Having the ability to detect oil spills is one of the first steps to eradicating them. If the presence of an oil spill is unknown, nothing can be done to stop it or slow its spread throughout the ocean. The most efficient way known to detect oil spills is with remote sensing.

The purpose of this paper is to present the results of the study of remote sensing detection of oil spills. There are several remote sensing sensors and instruments that are able to detect the presence as well as environmental effects of accidental oil spills. This study specifically analyzes data retrieved from sensors provided by NASA to analyze the Deepwater Horizon Spill. Data-processed images supplied by NASA from the year before the oil spill and the time frame of the oil spill are the basis of this study.

2. Deepwater Horizon Oil Spill

The Deepwater Horizon Oil Rig located in the Gulf of Mexico exploded on April 20th, 2010. The explosion killed 11 men and injured 17. In addition to human lives being compromised, the marine life and plant life in the surrounding area suffered greatly for months. The oil spill also endangered some species of birds and turtles native to the area. Oil causes animals to drown, die from hypothermia, become hydrated or even constrict their movement, causing them to die. Those animals that did survive the effects of the oil spill now have to live in a destroyed habitat. These consequences of the oil spill were immense due to the fact that this spill is the largest oil spill since the beginning of human extraction of oil. On July 15th, the oil spill was finally stopped after it had already leaked 4.9 million barrels of crude oil. The oil rig was owned by the oil and gas company, British Petroleum, more often referred to as BP. Therefore, BP was claimed as responsible for the cost of damage as well as the cleanup of the oil.

The removal of the oil spill was not as simple as with other oil spills due to its size. The original plan was to burn large portions of the oil away. However, because of the size, conditions, and the amount of environmental damage expected, the burning option was not executed until later. When conditions allowed burning of the oil, this method was used as well. Since burning of the oil was not available at the beginning of the oil spill, BP employed numerous machines remove the oil. These machines were designed to separate oil from water, which greatly helped to reduce the amount of oil. Skimmers were also originally banned at the start of the oil spill. Skimmers are small devices that float over areas with oil and extract the oil from the water. but later a skimmer was brought to the site without any significant results. The most interesting method of removing the oil was the oil eating microbes which were genetically engineered to digest oil. Unfortunately, the effects of this effort were not significant. In addition, since these microbes are a form of bacteria, it has been announced that the microbes are the possible cause of strange skin rashes found in people that live in the area. The most effective methods of removing the oil from the Gulf of Mexico were the burning of the oil and the separation of oil and water via machines.

3. Instruments and Parameters

The source of data for this experiment is NASA's online application, Giovanni, that allows anyone to analyze visual representations of remote sensing data. In order for this site to be useful, some level of understanding of the sensors and parameters is necessary. Giovanni has multiple interfaces, called instances that are grouped by scientific community. The instance that specifically applies to this study is ocean color radiometry. Within this instance there are multiple sensors which are each capable of measuring varying parameters. The four sensors which are used in this experiment are Sea WiFS at 9km, MODIS-aqua at 9km, MODIS-aqua at 4km, and MODIS-terra at 9km. The four parameters of interest for this study are: chlorophyll *a* content, colored dissolved organic matter, reflectance, and sea surface temperature. The Sea WiFS and MODIS-aqua instruments both have the capability to measure chlorophyll *a* content, colored dissolved organic matter (CDOM), and reflectance at 443nm. In addition to those three parameters, MODIS-aqua has the ability to measure the sea surface temperature, along with MODIS-terra.

These four parameters were chosen for this study to confirm the hypothesis that they indicate an oil spill. One hypothesis is that in the presence of an oil spill, the chlorophyll *a* content increases. One particular research experiment "did not find significant differences in the chlorophyll *a* concentration of the polluted and reference sediments" (Riaux-Gobin). However, in another experiment that deals with the effects of oil on chlorophyll a content, results showed that oil was a cause of increase in chlorophyll *a* content (Fabregas). For this discrepancy in the effect of oil spills on chlorophyll *a* content, this parameter is included in this study to try and retrieve more conclusive data about the relationship between chlorophyll a content and the presence of an oil spill. Another hypothesis is that the amount of colored dissolved organic matter also increases. Another experiment found that "the dissolved organic matter concentration of the ground water increased to 50mg C L^{-1} adjacent to the oil spill" (Ryan). Also, the reflectance should be higher in the area of an oil spill due to the physical properties of the oil. Oil, especially on the surface of the water, has sheen to it, causing the sunlight to be reflected back into the atmosphere and towards the satellites. Sunlight penetrates the surface of clean ocean water, therefore having a lower reflectance than that of an oil spill. As a result of an increase in reflectance, the sea surface temperature should be lower in areas of an oil spill and higher in the surrounding areas of clean ocean water. Since oil spills reflect the sunlight and reflectance and absorption are opposites of each other, oil spills do not absorb the sunlight, hence resulting in a lower temperature.

Through the Giovanni website the information can be arranged in several different ways. For this study, time-averaged Latitude-Longitude maps are produced from data from these parameters and instruments. These maps show the data in the form of a color scale over the actual map of the area. The dark purple color indicates the least concentration of that particular parameter, while the red color indicates the highest concentration of the parameter. Each map needs to be interpreted differently depending on the parameter and the highest level of concentration listed for that image. This type of map was chosen for this study because we want to compare the location as well as the varying levels of concentration of the parameters of interest. These maps display the information in a much more meaningful way rather than observing a table of values of concentrations of each parameter.

4. Results

The time-averaged Latitude-Longitude maps generated from the Sea WiFS at 9km, MODIS-aqua at 9km, and MODIS-aqua at 4km, MODIS-aqua at 4km for each specific parameter, area, and time frame, were very similar to each other. However, the MODIS-aqua at 4km seemed to give the best image each time, so most of the images provided are from the MODIS-aqua at 4km. The sea surface temperature parameter was the one parameter that was not available on MODIS-aqua at 4km. It was only available on MODIS-aqua at 9km and MODIS-terra at 9km. MODIS-terra at 9km gave the best results for the sea surface temperature. The results discussed in this paper consider all of the maps from each of the sensors, but for visual purposes, the images provided for each parameter will only be from one sensor.

The time-averaged Lat-Lon maps for the chlorophyll *a* content were generated and analyzed from the data from the instrument MODIS-aqua at 4km. For the desired time frame, the only maps able to be produced were from select months from the year before and the year of the occurrence of the oil spill: April 2009, April 2010, May 2010, and June 2010. When comparing April 2009 and April 2010, it is clear that there is more chlorophyll content near the site of the explosion of the oil rig as well as towards the shore closest to the oil rig. The maps from May 2010 and June 2010 show higher concentrations of chlorophyll *a* near the shore closest to the oil rig, but not as much near the actual oil rig. The maps also display higher concentrations of oil between the site of the oil spill and the coast of Florida.

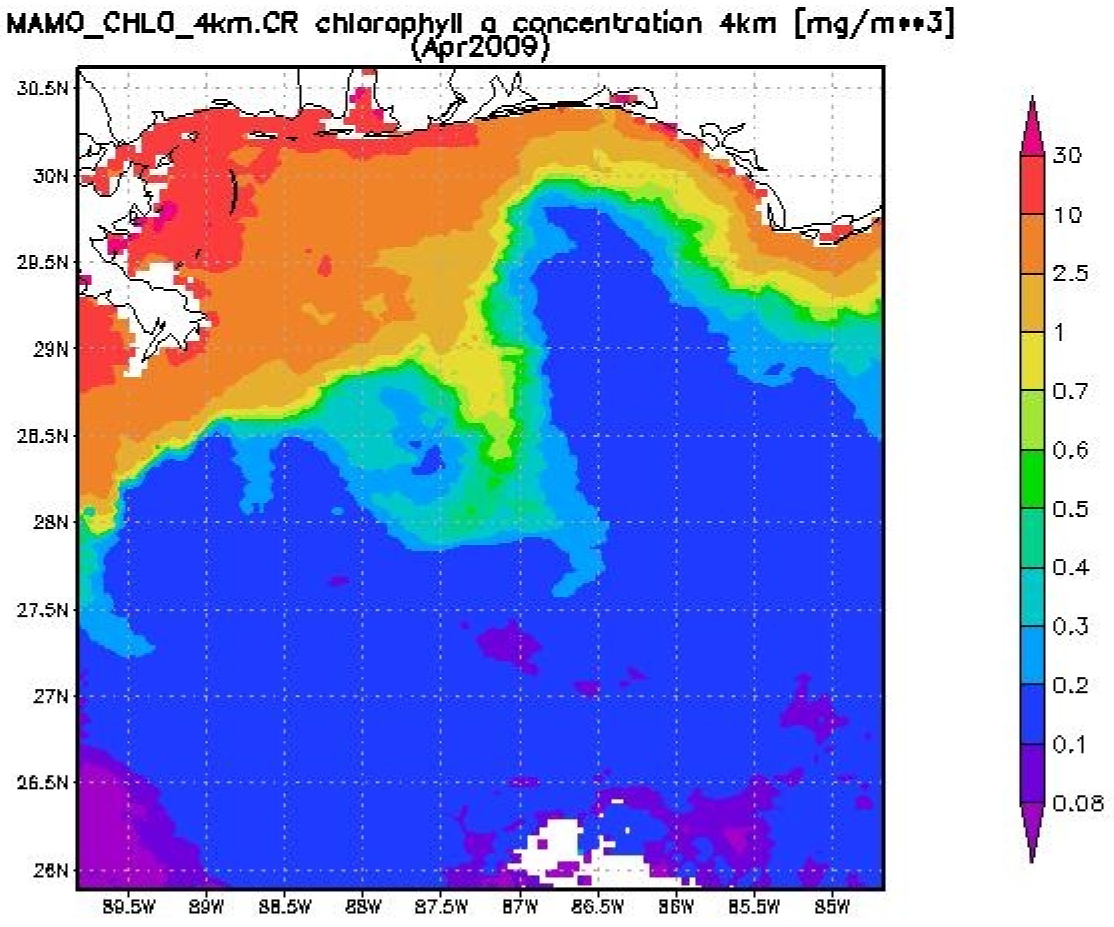


Figure 1.1: Chlorophyll a concentration for April 2009.
Instrument: MODIS-aqua at 4km

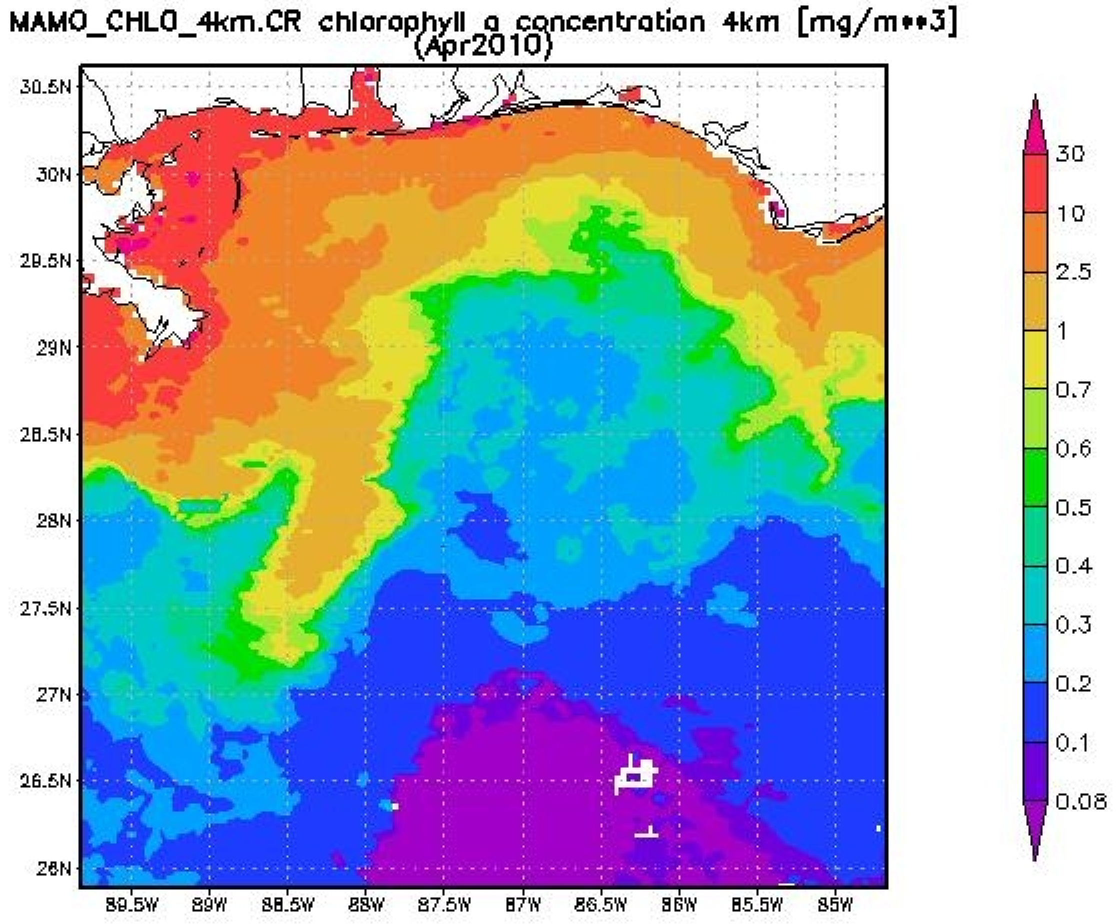


Figure 1.2: Chlorophyll a concentration for April 2010.
Instrument: MODIS-aqua at 4km

The time-averaged Lat-Lon maps for the sea surface temperature were generated from data from the MODIS-terra instrument. Maps were created for each month, April to June, for the year before oil spill and the year the oil spill occurred were generated. Comparing April 2009 and April 2010, it is evident that the sea surface temperature off the coast is cooler in 2010 than it was in 2009. When comparing the two maps from May, the same result was evident. However, the overall temperatures in May were higher than they were in April for both years. When looking at the maps generated from June for both years, the cooler temperatures were slightly farther away from the coast in the year 2010.

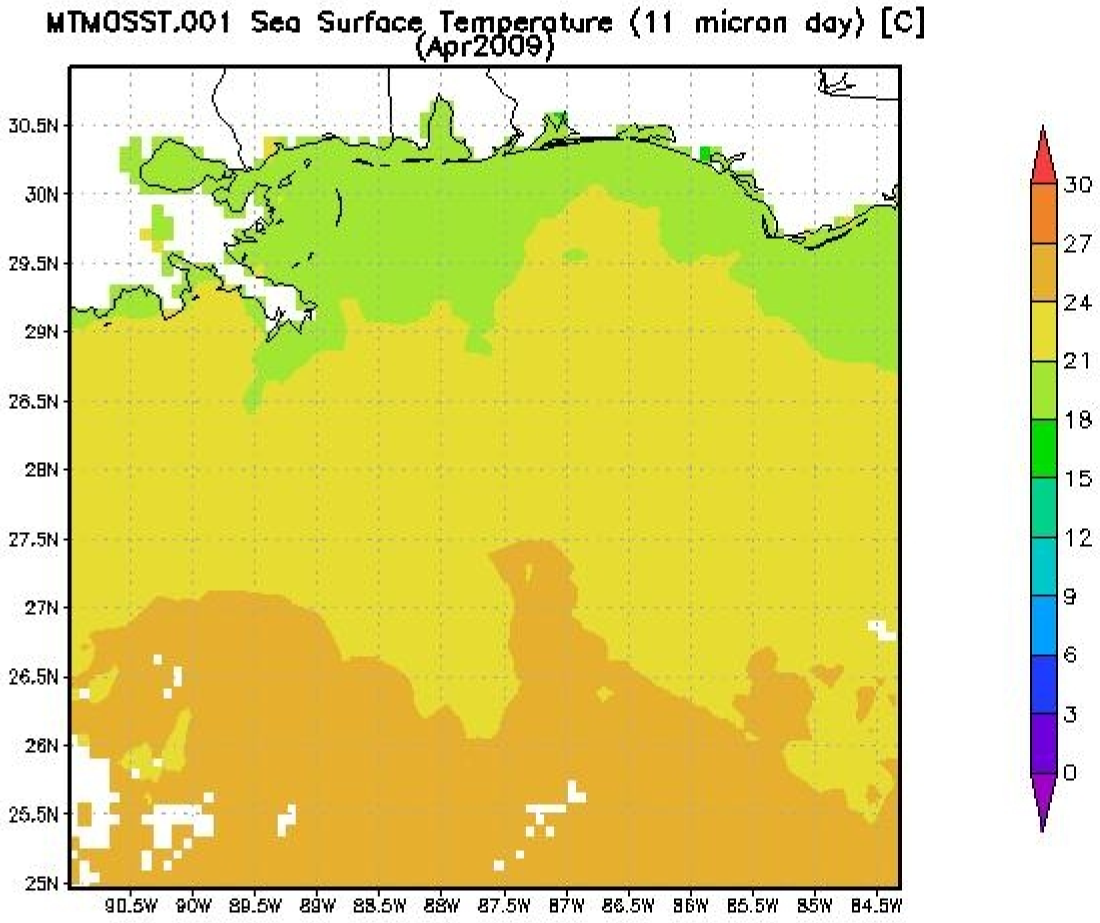


Figure 2.1: Sea surface temperature (day) for April 2009.
Instrument: MODIS-terra at 9km.

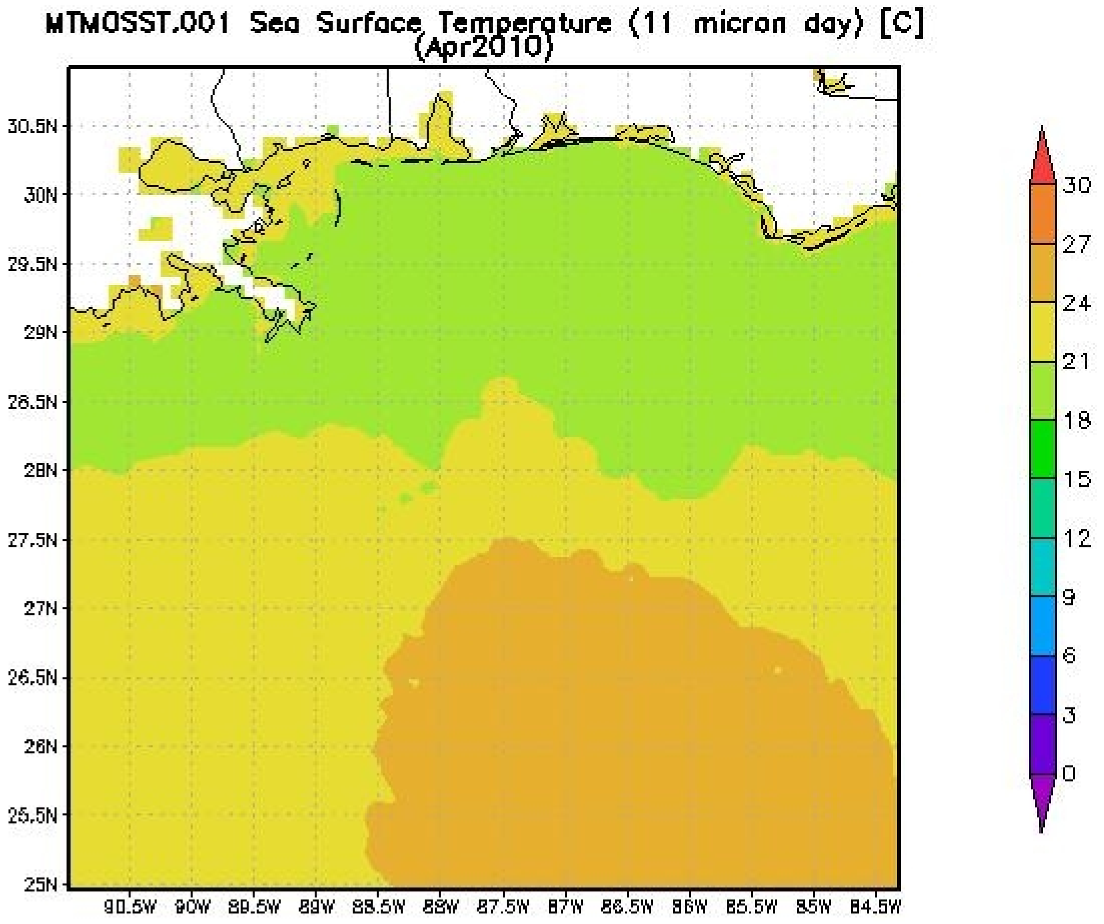


Figure 2.2: Sea surface temperature (day) for April 2010.
Instrument: MODIS-terra at 9km.

Reflectance at 443nm also had Lat-Lon maps generated from the MODIS-aqua at 4km. The two maps for April show less reflectance near the shores and more reflectance towards the middle of the ocean. The maps for May 2009 and June 2009 were not available, but the maps for May and June of 2010 showed much brighter reflectance a small distance off the coast than in the map for April.

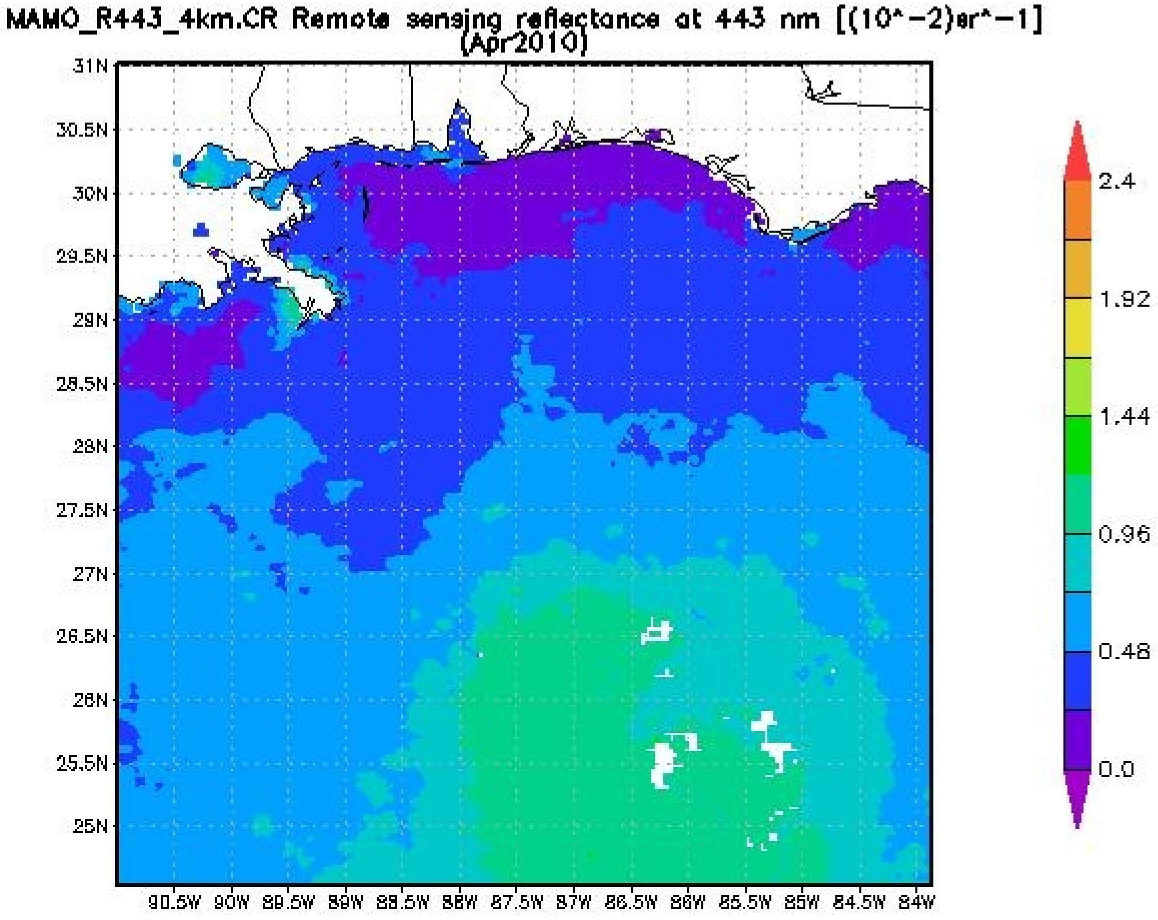

MAMO_R443_4km.CR Remote sensing reflectance at 443 nm [(10^-2)sr^-1]
(Apr2010)
31N
30.5N
30N
29.5N
29N
28.5N
28N
27.5N
27N
26.5N
26N
25.5N
25N
90.5W 90W 89.5W 89W 88.5W 88W 87.5W 87W 86.5W 86W 85.5W 85W 84.5W 84W
2.4
1.92
1.44
0.96
0.48
0.0

Figure 3.1: Reflectance at 443nm for April 2010.
Instrument: MODIS-aqua at 4km.

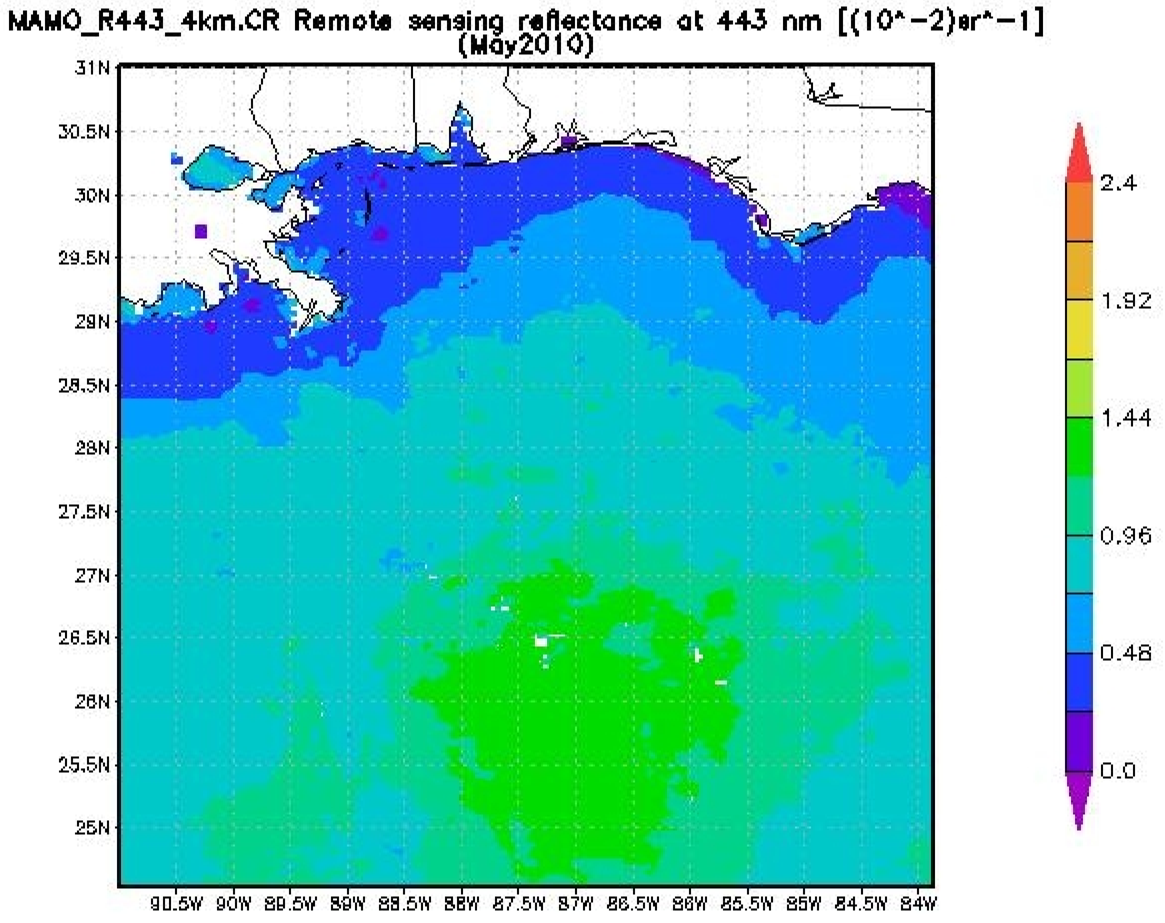


Figure 3.2: Reflectance at 443nm for May 2010.
Instrument: MODIS-aqua at 4km.

Lat-Lon maps were generated from the parameter CDOM, or colored dissolved organic matter from data retrieved from the instrument MODIS-aqua at 4km. April 2009 shows higher concentrations of dissolved organic matter along the coast. April 2010 shows a similar pattern, except concentrations near the coast are not nearly as high as that of April 2009. However, the overall concentration of dissolved organic matter was much lower in 2010. The maps for May 2009 and June 2009 are unavailable, but the maps from May and June of 2010 show significant less concentrations of dissolved organic matter. June 2010 showed almost no dissolved organic matter at all in most of the region.

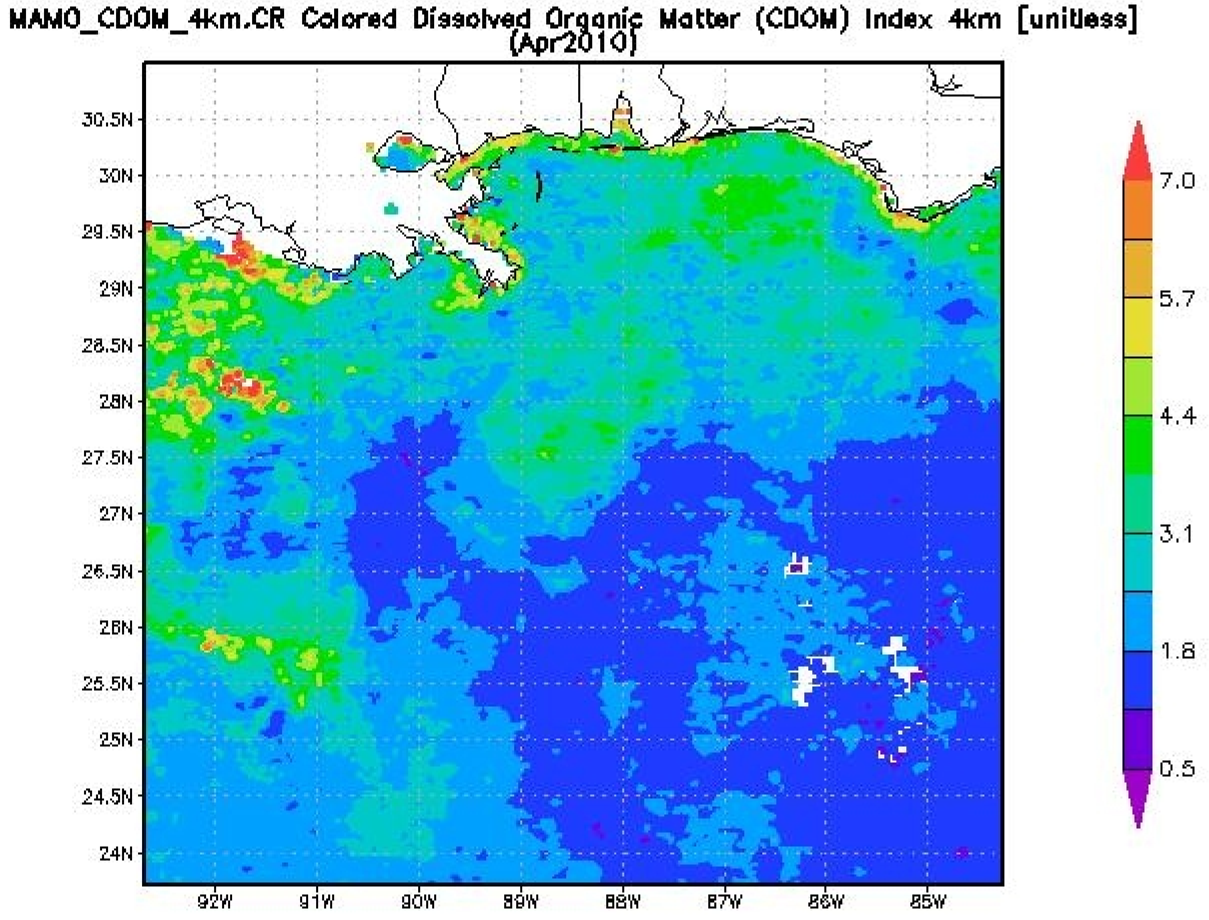


Figure 4.1: Colored dissolved organic matter for April 2010
Instrument: MODIS-aqua at 4km.

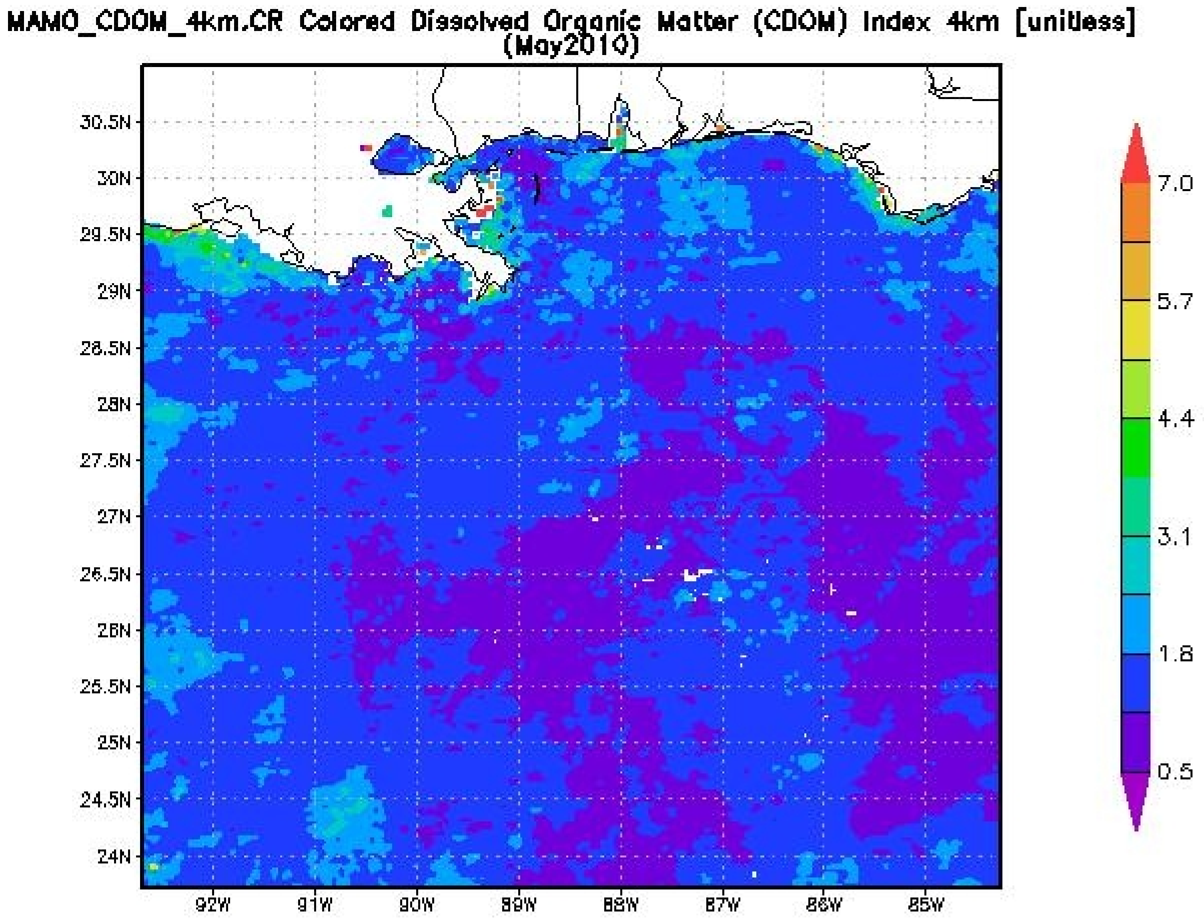


Figure 4.2: Colored dissolved organic matter for May 2010.
Instrument: MODIS-aqua at 4km.

5. Discussion

Most of the results of this experiment were consistent with the hypothesis. The images above are a few of the images from each parameter that were part of this study.

The images displaying the chlorophyll *a* concentration showed higher concentrations near the oil rig at the beginning of the spill, then in later months, the chlorophyll concentration increased more toward the shore and traveled east towards the shores of Florida. This pattern matches the movement pattern of the oil due to the ocean currents and winds. Therefore, it can be concluded that the hypothesis that chlorophyll *a* concentration is higher in the presence of oil.

The maps generated for sea surface temperature display cooler temperatures in the area of the oil spill, just as predicted. Both April and May showed cooler temperatures towards the coast which is where the oil spill was located during these months. The images provided demonstrate that the sea surface temperature is indeed cooler where the oil spill is located. June 2010 showed slightly cooler temperatures farther away from the coast because the oil spill had diminished, spread out, and traveled farther away from the shores.

The maps of the reflectance show brighter colors in the areas of the oil spill because the oil spill reflects more sunlight than the surrounding ocean. The images available show the first two months of the oil spill. As you can see, the overall reflection in June 2010 is higher than in May 2010, especially in oil-ridden areas. The black color of the oil would normally absorb sunlight. However, in this case, the physical properties of the oil cause it to be more reflective than the clean, oil-free ocean surrounding it. This is consistent with the results of the study which display that the presence of oil and reflectance of that area are positively correlated.

The parameter that did not give results as confirming as the others is the colored dissolved organic matter. This could be due to the efforts to remove the oil, so therefore the amount of dissolved organic matter is lower in the months of May and June. Another possible reason for the lack of results from CDOM is that it might be at the wrong wavelength. Another sensor might be able to detect the increase in CDOM at a different wavelength. The dispersants released in the affected area might interfere with the satellite information, causing the CDOM to decrease rather than increase. Further investigation into this parameter would give more conclusive results.

Overall, this experiment was successful in choosing the appropriate sensors in combination with the parameters. It can be concluded that chlorophyll, sea surface temperature, and reflectance are all factors that alter due to the presence of an oil spill. The chlorophyll a content and reflectance both increase with the presence of an oil spill while the sea surface temperature is actually lower over the oil spill. The colored dissolved organic matter is a parameter in which other experiments might find different results due to the instruments and the information available.

6. Future Research

This study is just a beginning to what could be a very advantageous experiment. If this study were to continue, there are many of different factors to consider. More data would need to be collected in order to make the experiment more reliable.

I would explore other sensors available, not just from the Giovanni website. These other sensors might have access to information about other parameters that could contribute to detecting oil spills.

Alternate locations of interest would be analyzed to collect more data and possibly confirm that the four parameters used in this study actually detect oil spills. This would eradicate the possibility results that were coincidental. Other large oil spills from history would be investigated using the same parameters from the same instruments as well as any extra that might produce results. Some other possible locations to study would be any other significant oil spills that occur or any oil leaks produced by crude tankers or ships.

Using *in situ* measurements, measurements taken on site, of areas of current oil spills would be able to confirm the data from the satellites. However, in situ measurements would only be possible to collect from future oil spills.

Overall, this study could potentially be very advantageous. After extensive research, scientists could use these satellites to detect any oil spills that are undetectable by the human eye. If these usually undetected oil spills can be detected early on, measurements can be taken to start removing the oil from the environment immediately. This could potentially save many of marine and plant life that are often affected by oil spills.

References

Fabregas, J., Herrero, C., Veiga, M. (1983). Effect of Oil and Dispersant on Growth and Chlorophyll a Content of the Marine Microalga Tetraselmis suecica.

Riaux-Gobin, C. (1993) Long-term changes in microphytobenthos in Brittany estuary after the 'Amoco Cadiz' oil spill

Ryan, J.N., Aiken, G.R., Backhus, D.A. Vilholth, K.G., and Hawley, C.M. (2000) Investigating the Potential for Colloid- and Organic Matter-Facilitated Transport of Polycyclic Aromatic Hydrocarbons in Crude Oil-Contaminated Ground Water.

e-Research: A Journal of Undergraduate Work, Vol 2, No 2 (2011)

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES

Home > Vol 2, No 2 (2011) > **About the Contributors**

Andrea Cyr graduated with a BS in Biological Sciences in Spring 2011. She was an Amgen Scholar and conducted this research at the Laboratory of Genetics, The Salk Institute of Biological Studies, La Jolla, CA. She plans to attend medical school. Her faculty mentor was Matthew Weitzman.

Matthew Shaffer is a Math/Computer Science major. His faculty mentor was Atanis Radenski in the School of Computational Sciences.

Alexander Barrett graduated with a BS in Mathematics in Spring 2011 and has begun pursuing an MS in Computational Sciences at Chapman. His faculty mentor was Cyril Rakovski in the School of Computational Sciences.

Dung Bui is majoring in Mathematics. His faculty mentor was Mohamed Allali in the School of Computational Sciences.

Amanda Kristedja is pursuing a double major in Physics and Computational Science. She plans on attending graduate school in engineering. Her faculty mentor was Hesham El-Askary in the School of Earth and Environmental Sciences.

Mohamed Allali is Associate Professor of Mathematics and Computer Science, School of Computational Sciences, Schmid College of Science and Technology, Chapman University.

Cyril S. Rakovski is Assistant Professor of Mathematics and Computer Science, School of Computational Sciences, Schmid College of Science and Technology, Chapman University.

Matthew Weitzman is Associate Professor and Pioneer Developmental Chair, Laboratory of Genetics, The Salk Institute for Biological Studies, La Jolla, CA.