Real-time Adaptive Morphing Website Modeled Per User and Optimized Across Users

by

Christopher J. Perciballi

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 2010

[June 2010]

Author_____
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by_____
Glen L. Urban
David Austin Professor of Management Science
Chairman, MIT Center for Digital Business
Thesis Supervisor

Accepted by_____
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Real-time Adaptive Morphing Website Modeled Per User and Optimized Across Users

by

Christopher J. Perciballi

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Morphing is a powerful tool for providing users with information in a format that benefits them most. It has been shown to increase trust and sales. This thesis describes the implementation of a modular website that morphs based on the click stream of each individual user and learns how to pick the optimal morph based on aggregate user results. The main components are the website controller, the Bayesian Inference Engine, and the Gittins' Optimization Engine. The website controller acts as the interface between the user input and the mathematical modeling of the user's cognitive styles. It uses the Bayesian Engine to update the model and the Gittins' Engine to select the best morph in order to modify the website view. The project was run in survey format to test the effectiveness of morphing for the Suruga Card Loan advice site as well as to test performance and feasibility of real-time morphing and optimization.

Thesis Supervisor: Glen L. Urban
Title: David Austin Professor of Management Science
Chairman, MIT Center for Digital Business

Table of Contents

List of Figures

1 Introduction

This thesis describes the design and implementation of a modular and scalable website that performs cognitive and cultural style morphing based on user click stream data. The morph decision process is optimized at the level of individual users as well as across all users. First I will describe the motivation for morphing and previous work done in the field by Professor Urban and his team. Then in Section 3, I will give an overview of the system and its functionality. In Section 4 I will describe the implementation of the major system components. Section 5 describes the results of the complete system, followed by contributions in Section 6 and future work in Section 7.

2 Motivation and Background Work

In this section I will give a high-level description of morphing motivation and theory, as well as previous work in the field. For more information on morphing motivation, please refer to "Morph the Web to Build Empathy, Trust, and Sales" by Glen L. Urban, John R. Hauser, Guilherme Liberali, Michael Braun, and Fareena Sultan. For a more detailed explanation of previous work, please refer to "Website Morphing" by John R. Hauser, Glen L. Urban, Guilherme Liberali, and Michael Braun.

2.1 Motivation

Morphing has been shown to increase sales by building empathy and trust. Through morphing, users are presented with information in a way that best matches their cognitive style, allowing them to process the information more easily. Imagine trying to explain how a circuit works to a classmate and then to a younger sibling. The two explanations would be likely to vary in content, vocabulary, and depth. The same ideas apply when trying to create a marketing advisor. Some users are more visual and will appreciate graphs and figures. Others are more verbal or technical and will gain more from statistics and

descriptions. By adapting the advice website, companies can appeal to more users and thus increase trust and sales.

2.2 Previous Work

The British Telecom advice website, as described in "Website Morphing", was the first site to make use of the morphing ideas and algorithms. It used a Bayesian Inference Loop and a dynamic programming Gittins' solution to pick the optimal morph. The project demonstrated the steps to making a morphing website and provided the foundation for the Suruga Card Loan site. In order to create the site, a priming study was used to determine the cognitive styles of users and match them to the preferred morphs. This data was used to classify the morphs and links in the website so that cognitive styles could be inferred through click stream data.

3 Project Overview

The system described in this thesis performs cognitive morphing as well as logging clicks and statistical data to be analyzed for research purposes. The main components, as depicted in Figure 1, are the Bayesian Inference Engine, the Gittins' Optimization Engine, and the Content Management System. After each click, the Bayesian Engine is run to update the estimate of the user's cognitive style. After a fixed number of clicks, as set after the previous study, the Gittins' Engine is run to determine whether to change the morph. The morph number selects a row from a morph translator array, which is used to determine which visual components are loaded in the HTML page.

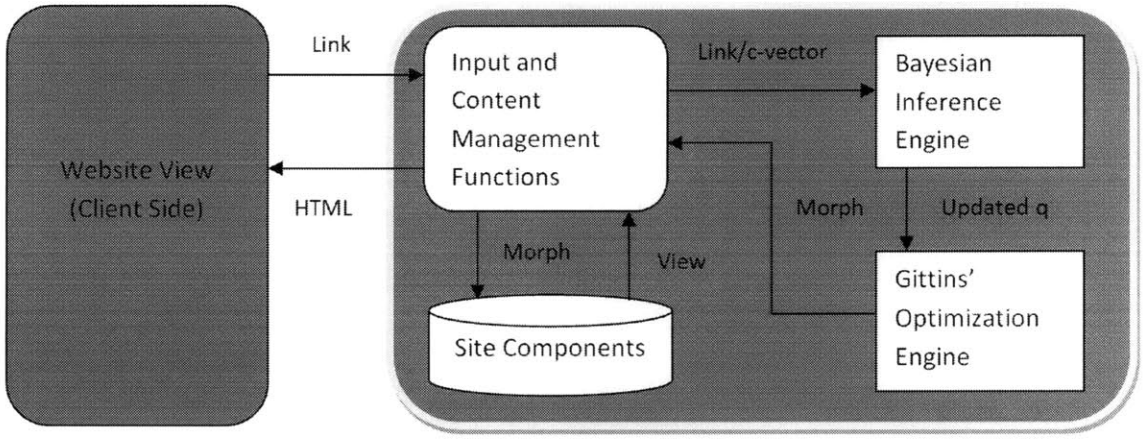

Figure 1. High level diagram of system components.

The site is also required to interface with a survey panel, which sent users to the site after collecting responses to priming questions. In order to facilitate the interface between the survey panel and the site, a separate landing page accepts and redirects users to the Suruga site by opening another frame. The landing page is also used to determine "success" or "failure", defined as the user's wish to receive more information about card loans, the product being described. Users could click one of two buttons: one to get more information then leave the site, or one to leave the site without information. Closing the frame without clicking on one of these buttons was also a failure, and the landing page was used to handle that case.

3.1 System Functionality Walkthrough

This section describes the path users take through the site and outlines the roles of site modules in the process.

First, users are directed to the landing page from the survey panel. The landing page has instructions and leads to the page that launches our survey setup in a new window, shown in Figure 2. The window displays the Suruga Card Loan page in a frame and keeps a timer to ensure that the user browses for a sufficient period of time. The page has two buttons at the top for leaving the page with or without

getting more information about a card loan. These buttons help determine whether the user should be counted as a success or failure.

Figure 2. The Suruga Card Loan site displayed in a frame. The buttons at the top right allow the user to leave the site with or without more information on getting a card loan.

When the Suruga Card Loan page is first loaded, a random morph is selected. In addition, control users are randomly chosen to see one morph for the duration of their visit to the site. We will focus on the case where morphing is enabled. When the user clicks on a link, the function for the page being viewed is called. Many of the links on the site have been assigned cognitive style vectors, c-vectors, based on previous surveys and observation. The vectors, which contain weights for various dimensions, are used by the Bayesian engine. Whenever such a link is clicked, the Bayesian Engine is run to update the model for the user. If the selected link does not have a c-vector, the click is still recorded and relevant values are updated. These values include the total number of clicks and clicks per morph. On every fifth click, the Gittins' Optimization function is called. Based on previous studies, it was determined that on average deciding whether to morph every fifth click was optimal. This allowed us to gain a performance advantage. If the function determines that a morph change would be beneficial, the new morph is set and the next page (the page just selected by the link) will reflect the changes.

After a time period defined by survey requirements has elapsed, the user can decide to leave the site. If the user elects to receive more information about card loans, they are asked to fill out a short questionnaire. This sets a parameter to success in the landing page database. If the user elects not to

receive more information or simply closes the browser, the parameter is unchanged and signifies failure. Back on the landing page, pressing continue calls a function on the Suruga Card Loan site that updates the alpha and beta distributions based on the success or failure. The user is then redirected back to the survey panel for closure.

4 Implementation

This section details the implementation of main site components as well as their interaction in the morphing process. The site is implemented in Cake and PHP with MySQL for storage of user data and morphing parameters. The Bayesian Engine and Gittins' Optimization Engine are separate PHP modules that we treat as black boxes. The website controller is the interface between user input and the black boxes. Figure 3 below depicts the flow of the morphing process and the interaction of the black boxes with the controller.

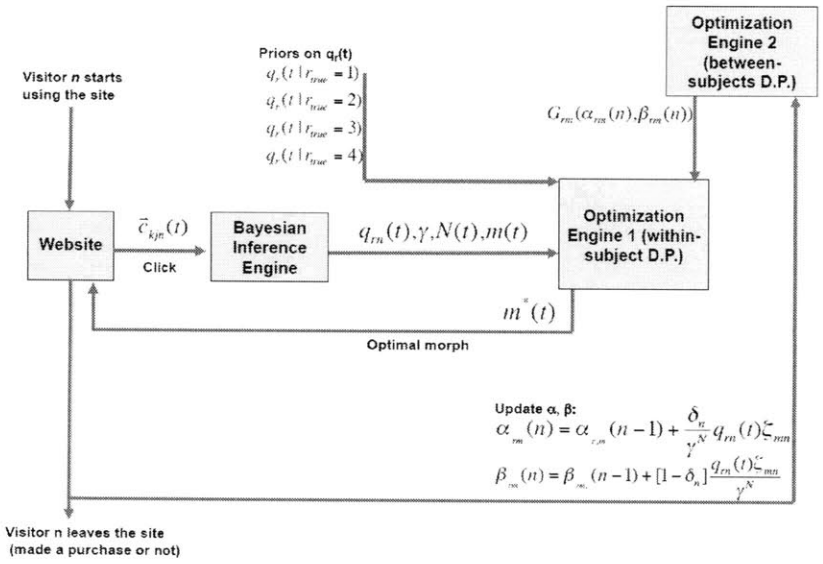

Figure 3. Architecture of system and black boxes by Guilherme Liberali. Clicks from the website go through the controller interface to invoke the Bayesian Inference Engine and the Gittins' Optimization Engine. Success and failure events update the alpha and beta distributions to improve morph selection across users.

4.1 Site Workflow

When a user clicks on a link, the page function for the page that the link points to is called in the controller. Pages with c-vectors call the math_engine() function, which reads the c-vector of the link and all other links on the previous page and runs the Bayesian engine. For example the two image links on the home page, shown in Figure 4, have c-vectors. Clicking on one invokes function level1($link, $originPageName). The parameters $link and $originPageName are passed through the URL. The picture of the house links to http://advocacy.mit.edu/cp/suruga_web_pages/level1/3/page0, where 3 is the link and page0 signifies the home page as the origin. The math_engine() function uses the page and link information to look up the c values and construct vectors using makeCVector(). Then it runs the Bayesian function, which returns the array of updated q values.

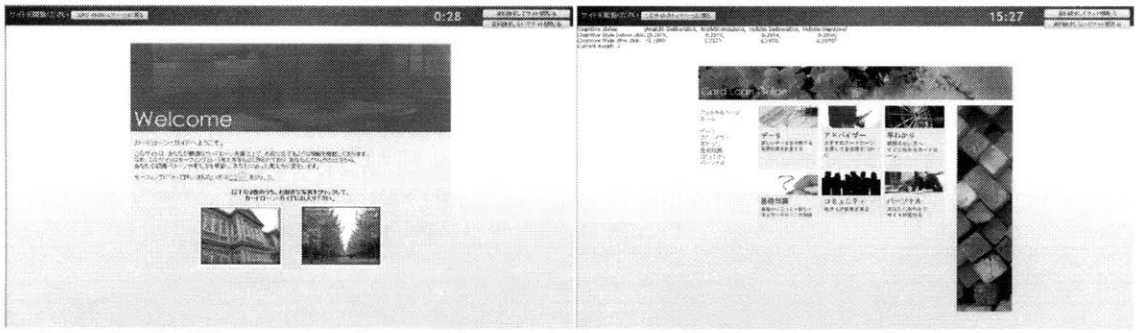

Figure 4. Results of clicking on a link. The cognitive style before and after the click as well as the morph number are displayed at the top of the page.

After every fifth click, math_engine() also calls the Gittins' function Optim_engine() to select the best morph. If the morph changes, math_engine() increments the number of morph changes and stores the new morph in the session variable.

The page functions select page elements by calling the function load_layout_rules(). This function modifies the morph array, which is read by the template site pages. This allows the pages to be built from components rather than having a different version of each page for each morph.

To update alpha values in the event of a success, the function record_alpha() is called by opening the record_alpha page of the site. Beta values are updated using record_beta(). The landing site

opens these pages during the survey, and the pages redirect immediately to the survey site so the process is transparent to the user. The functions take the user's ID as a URL parameter and use the ID to find the last click log in the suruga_user_statistics database. The click log from the most recent click contains the most recent q and zeta vectors, which are needed to update alpha and beta. The update_indices() function in the Gittins' Optimization module is used to update the values.

4.2 Controller

The controller contains initialization functions as well as a function for each page in the site, which are called before the page view is constructed and displayed. The controller first initializes the Ω matrix and q_o, the initial cultural and cognitive style estimate vector. Based on the previous study, we were able to define q_o = (0.1876, 0.2575, 0.2954, 0.2595) to best approximate the user's cognitive style. The controller keeps track of the number of morph changes, the number of total clicks made by the user, and the number of clicks per morph. Clicks per morph are stored in the zeta array, defined as clicks per morph divided by total clicks.

The function index() is called when a user goes to the Suruga Card Loan home page. The function initializes all of the variables used to conduct the survey and to run the black boxes and display the correct morphs. If it is the user's first time to the home page, the user's ID is set. For survey purposes, it randomly assigned some users to the control group by fixing the morph. The variables that are initialized are: the total number of clicks on the site, the zeta array, and the number of morph changes.

The controller contains a function for each page on the site. If the page has an associated c-vector, the function calls math_engine(), which acts as an interface between the controller and the black boxes. It updates the state variables and runs the Bayesian and Gittins' functions as needed. If the page

does not have a c-vector, the function calls save_click(), which updates the state variables accordingly but does not run the Bayesian or Gittins' functions.

The math_engine() updates the total number of clicks and the number of clicks per morph and logs user statistics in the database suruga_user_statistics. The function uses the click information from the user to look up the c-vector associated with the selected link as well as the c-vectors associated with all other available links on the page. These vectors are the input to the Bayesian engine, which math_engine() calls to update the model of the user's cognitive and cultural style.

The math_engine() function also provides the input to the Gittins' Optimization function, which was redesigned after previous field studies. First, the function reads the alpha and beta values from the database. A pair of values (α,β) is retrieved for each combination of *r* and *m* values, where *r* is the vector of cognitive style groups and *m* represents morphs one through four. Thus there are 16 (α,β) pairs. The alpha and beta values are then used as indices into the G matrix, a 3000 by 3000 matrix of values. The Gittins' Optimization function takes as input the current morph number, the number of clicks, the q vector, and the G values determined by alpha and beta.

The controller also contains functions to update alpha and beta values for success and failure. These functions are called by the landing page rather than by user clicks. After the update is complete, they redirect back to the landing page. The alpha and beta values are updated according to the following formulas:

$$\alpha_{r,m}(n) = \alpha_{r,m}(n-1) + \frac{\delta_n}{\gamma^N} \mathrm{q_{rn}(t)\zeta_{m,n}}$$

$$\beta_{r,m}(n) = \beta_{r,m}(n-1) + \frac{(1-\delta_n)}{\gamma^N} \mathrm{q_{rn}(t)\zeta_{m,n}}$$

where δ_n is 1 for success and 0 for failure, γ is a weighting value defined to be 0.99 in our system, N is the total number of morph changes for the user, and ζ is the zeta vector with elements click per morph divided by total clicks. These values are modified in the database and provide the basis for optimization across all users. The updated alpha and beta values are used to index the G matrix. As a result, the updated values will impact the Gittins' Optimization function for subsequent clicks made by all users on the site.

4.3 Bayesian Inference Engine

The Bayesian Inference Engine uses Bayes' Rule to update the estimate of the user's cognitive and cultural style based on the link the user clicks. It takes as input the c-vector from the link selected as well as the c-vectors from all other available links on the page. Based on priming studies done on the site, we can compute the probability of the user clicking on a link given his or her cognitive style. Using Bayes' Rule, we can compute the posterior distribution for cognitive style given that they clicked on the link.

4.4 Gittins' Optimization Engine

The updated Gittins' Optimization function was designed and implemented in R by Guilherme Liberali before we ported it to PHP. The PHP code can be found in Appendix A. It performs backwards induction to determine the optimal morph to show the user. The new function makes use of the updated Gittins' Index table and *r-m* table. The optimization function takes as input the current q vector, the click number, the relevant G values, the system parameter gamma, and four fixed empirical q vector values. The G values are extracted from the 3000 by 3000 matrix of G values using the alpha and beta values from the *r-m* table, which is stored in MySQL for global access by all user sessions. The system uses four values for *r* and four morphs, making G a length 16 vector.

4.5 *r-m* Table Implementation

The *r-m* Table stores α and β values for each combination of *r* and *m*, where *r* is the set of cognitive styles and *m* is the set or morphs. The values are used to track success and failure of the morphing

system and determine the optimal morph in the Gittins' Optimization function. In the British Telecommunication site, $r = 64$ and $m = 64$, resulting in nearly 10-second lookup times in a nested for loop on r and m. To alleviate this performance problem, the table was stored in local memory in each user session. In this system, the r-m Table must be global in order for optimization to be carried out across users. Due to the decrease in number of morphs to four and resizing r to four, the table could be stored in a database without significant delays during lookups. The values are now stored in r_m_tables in the MySQL database.

4.6 Gittins' Index Table

The Gittins' Index Table, or α-β Table, stores the indices needed for running the Gittins' Optimization function indexed by (α,β) pairs. It is a 3000 by 3000 table of floats that must be accessed 16 times every fifth click on the site. The table is far too large to store in local memory, and causes performance decreases when stored in a database. Due to time constraints in getting the survey to field, we needed a quick solution. I decided to break the table up and store it in text files by row. The lookup function uses β to choose the correct file, then uses α to index into the file and return the correct value. The files are stored locally on the server and are named with the prefix "G_." For example, G_1.csv contains the first row of the table in comma delimited format. The code can be found in Appendix A.

4.7 Lessons Learned

The final site is a complex system with many interacting parts. The site was not originally designed to run the survey, and the landing site added some extra complication. However, at the same time, it provided functionality that would have been more difficult to implement otherwise. The timing function that ensured users browsed for a sufficient period of time was a simple addition to the landing page. This is because the frame that contains the Suruga Card Loan site remains unchanged during the browsing period. The mechanism for determining success and failure would have been a more natural fit in the Suruga Card Loan site itself, but it would have required modifying every page to insert the

buttons. It would have been much simpler if this had been planned from the start. The modifications required to implement the survey functionality demonstrate the well-known need for a good initial plan and design. The modular design of the system allowed for the necessary changes to be made.

In systems with many interacting parts, unexpected problems arise during field study. Though the system was tested with several complete survey responses before going to field, problems arose with tracking IDs between the landing page and the Suruga Site, and with memory limitations. The survey was run in batches of 200-300 users in Japan, which could lead to a wide variety of browsers and range of technical user knowledge. It was impossible to predict potential problems, so closely monitoring the site and data collection was critical. A major lesson from the data collections is to log all information possible, including creation time of all database entries, session ID for every click, and user ID for every click. This provides a strong possibility for recovery in the event of any problems collecting data.

5 Results

Overall, the site and survey were a success. Early results show that morphing led to an increase in purchase interest as opposed to the control group. Site performance was acceptable with the exception of two minor problems related to the database, described below. The site successfully morphed and updated the α and β values across users.

In order to correctly assign control groups, the home page makes a request to the MySQL table suruga_user_statistics, which logs each click made by all visitors to the site. Sorting the table has become very costly as the number of clicks has reached the thousands. After the survey completed, the home page took about 5 seconds to load. However, removing the sort returns the performance to normal, and a production morphing website would not experience this problem.

The α and β update presents a more important performance issue. Currently, the updated is handled when the landing page opens the Suruga Card Loan pages record_alpha or record_beta and

passes a user ID. The site then finds the last click made by the given user ID in the suruga_user_statistics table to find the clicks per morph and cognitive style estimate q, which are needed to perform the update. As the number of clicks by user on the site grows, the filter and sort time could become very expensive. In a large-scale production site, this may need to be addressed. One solution would be to keep a database with a profile of each user containing the necessary information, which would reduce the lookup to a filter problem only. This would keep performance reasonable as filtering by ID is very fast compared to sorting by creation time.

6 Contributions

This thesis demonstrates how to build and maintain a modular website that performs cognitive and cultural style morphing at the level of the individual user while learning how to pick the optimal morph from aggregate results. The black boxes have clean interfaces and are reusable in other systems. The modules may be accessed from outside the Suruga Card Loan site itself, which is useful for web projects. This allows the aggregate data to be updated seamlessly.

The project also provides an example of how a site can be used to collect research data without extensive changes to the underlying system. It was used to gain valuable data to evaluate the effectiveness of cognitive morphing and its feasibility for web applications, with the survey functionality coming from the landing site.

7 Future Work

The results of this project are very encouraging for future morphing projects. The modular black-box design allows the components to be applied to many applications. The Bayesian Inference Engine outputs a cognitive style estimate vector, and the Gittins' Optimization Engine outputs the optimal morph. In our system, these values are used to set a morph array, which determines which site components the user sees on a page. These values could be used for many other purposes. For example,

Professor Urban and his team are laying the groundwork for an ad morphing project in which the black-box outputs would determine which style ads a user site visitor would see.

The performance improvements will be necessary for the next project that is under consideration with Suruga Bank: a mobile morphing application. The mobile platform is especially exciting because it has not yet been attempted and there is great potential for increased utility. Due to the limited screen space and input functions on most mobile devices, format and presentation are critical to making a good application. Providing the user with the best display possible would likely lead to large gains in user satisfaction.

Bibliography

[1] John R. Hauser, Glen L. Urban, Guilherme Liberali, and Michael Braun. Website Morphing. Marketing Science, 2009.

[2] Glen L. Urban, John R. Hauser, Guilherme Liberali, Michael Braun, and Fareena Sultan. Morph the Web to Build Empathy, Trust and Sales. MIT Sloan Management Review, Summer 2009.

[3] Clarence Lee. User Adaptive Web Morphing: An Implementation of a Web-based Bayesian Inference Engine with Gittins' Index. Master of Engineering thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2008.

Appendix A. System Maintenance and Implementation Details

This section contains relevant code and implementation details for the system modules, as well as notes on maintenance and how to run the system.

Controller

The controller is implemented in suruga_web_pages_controller.php. It acts as the interface between the website and user input and the black boxes used for morphing. It contains a function for each page in the website and provides functions for updating alpha and beta.

The index() Function

The index() function is called when a user goes to the site homepage. It records the user's ID from the URL if no user ID exists. It initializes key user session variables: the q vector, morph number, number of morph changes, number of clicks per morph, and total number of clicks.

Page Functions

The controller contains a function for each page within the site of the form:

```
function level1($link, $originPageName) {
                $this->Session->write('link', $link);
                $this->Session->write('originPageName', $originPageName);
                $this->Session->write('last_seen_level', "page1");
                $page = $this->getPageModel($originPageName);
                if ($link != 5) {
                        $this->math_engine($page, $link);
                } else {
                        $this->save_click();
                }
                $this->set('SurugaWebPages', $this->SurugaWebPage->findAll());
                $this->pageTitle = 'Welcome to the Card Loan Advocacy System';
                $this->load_layout_rules(1);

                $this->my_history('home' , $this->Session->read('morph_array'));
        }
```

The most important role of these functions is to call either math_engine() or save_click(). The function math_engine() is called if the link has an associated c-vector and the Bayesian Inference Engine should be run, and save_click() otherwise. Descriptions of these functions can be found below.

Function math_engine()

The math_engine() function updates user data and logs the data in suruga_user_statistics. The logged data includes the user's ID, the page, link, morph before and after, q vector before and after, clicks per morph, alpha and beta values, and Gittins' Index values. The function calls run_bayesian_engine() to update the cognitive style model and it calls Optim_engine() to get the optimal morph

Function save_click()

The save_click() function was created to handle clicks on links that do not have c-vectors and should not trigger either of the black boxes. It updates the number of clicks per morph, and logs a click entry in suruga_user_statistics. It can be viewed as a placeholder for math_engine().

Site Content

The morph determines which elements will be displayed on the web page by selecting a vector from the morphTranslator array, which is defined as

```
var $morphTranslator = array(
                        array(),
                        array( "D1" => "0", "D2" => "0", "D3" => "1", "D4" => "1", "D5" => "0", "D6" => "0"), // 0
                        array( "D1" => "0", "D2" => "0", "D3" => "1", "D4" => "1", "D5" => "1", "D6" => "0"), // 1
                        array( "D1" => "0", "D2" => "1", "D3" => "0", "D4" => "0", "D5" => "0", "D6" => "0"), // 2
                        array( "D1" => "0", "D2" => "1", "D3" => "1", "D4" => "0", "D5" => "1", "D6" => "0"), // 3
        );
```

The vector entries point to different files and images in the content folders, and the PHP and HTML in the templates select the proper elements as shown below, taken from the template level1.thtml:

```
$morph = $session->read('morph_array');

if ($pic_array['reading'] == 'no') { $reading = 'x'; } else { $reading = (string)$morph['D1'];  }
                        if ($pic_array['analytic'] == 'no') { $analytic = 'x'; } else { $analytic = (string)$morph['D2']; }
                        if ($pic_array['deliberative'] == 'no') { $deliberative = 'x'; } else { $deliberative = (string)$morph['D3'];
        }
                        if ($pic_array['hiearchical'] == 'no') { $hiearchical = 'x'; } else { $hiearchical = (string)$morph['D4'];  }
                        if ($pic_array['individualistic'] == 'no') { $individualistic = 'x'; } else { $individualistic = (string)$morph['D5'];
        }
                        if ($pic_array['neutral'] == 'no') { $neutral = 'x'; } else { $neutral = (string)$morph['D6'];    }

                        $pic_loc = $reading.$analytic.$deliberative.$hiearchical.$individualistic.$neutral.$pic_array['file'];
                        $pic_loc = $svr_root_path.'suruga/level1/home/pics/'.$pic_loc;
                        $file_name = $pic_loc;
                        $img_str = '<img src="'.$file_name.'" width="140" align="right">';
```

Bayesian Inference Loop

The code to run the Bayesian Inference loop is found in Bayesian_engine.php. The function run_bayesian_engine($c_selected, $c_total) is called from math_engine(). It takes the c-vector of the selected link and the c-vectors of the other links on the page and outputs the updated q vector.

Gittins' Optimization Engine

The Gittins' Optimization Engine contains the function Optim_engine(), which perfoms the backwards induction to find the optimal morph. It also contains code to lookup Gittins' indices given α and β in the function lookup_gittins_index_new($alpha, $beta). The function update_indices($qrm, $clicks_per_morph, $total_morphs, $buy) takes q, zeta, total morphs, and delta and interfaces with the database to update α and β. This function is called from record_alpha() and record_beta() in the controller.

The new function for looking up Gittins' Indices is as follows:

```
function lookup_gittins_index_new($alpha, $beta) {
        if ($alpha < 1) {
                $alpha = 1;
        }
        if ($beta < 1) {
                $beta = 1;
        }
```

```
        if($alpha > 3000 || $beta > 3000) {
                return $alpha / ($alpha + $beta);
        }

        $G_floor = $this->lookup_G_helper(floor($alpha),floor($beta));
        $G_ceil = $this->lookup_G_helper(ceil($alpha),ceil($beta));

        return ($G_floor + $G_ceil) * 0.5;
}
```

The function uses the alpha and beta values to select the G files and index into them. It accounts for the fact that alpha and beta may not be whole numbers by using both the floor and ceiling of each and averaging the results.

Database Tables

The files to interface with databases are found in the app/models folder. The files used for morphing are: r_m_table.php, suruga_user_statistic.php, and suruga_page[0-7].php. The files with the prefix suruga_page contain the c-vectors for the links found on the given page. The databases can viewed from phpMyAdmin on the Advocacy server.

Problems Encountered

During the running of the survey, a few bugs arose. First, there was a problem with the site's memory allocation. The home page failed to load due to a PHP memory shortage. This was fixed by increasing memory_limit field in the php.ini config file.

There was a problem with some user IDs showing up as null or -1. This is believed to be a cookie problem as it does not occur in all cases and a wide variety of browsers may have been used by survey respondents. See the code in the controller's index() function for dealing with user IDs.