Quality and Reliability Engineering International

(wileyonlinelibrary.com) DOI: 10.1002/qre.2030

Published online in Wiley Online Library

Improving Websites' Quality of Service by Shortening Their Browsing Expected Path Length

Zohar Postelnicu, Tal Raviv and Irad Ben-Gal[*†]

This work proposes a method to improve the QoS provided to internet users by website servers. In particular, the goal is to minimize the expected number of browsing steps (clicks), also known as the expected path length, which are required to reach a website page by a community of users. We use Markov chain modeling to represent the transition probabilities from one webpage to another, and the first passage from page to page that can be retrieved from web server logs. The proposed method deletes links among webpages to minimize the expected path length of the website. Three different methods for achieving this goal are examined: (i) a greedy deletion heuristic; (ii) an approximated branch and bound algorithm; and (iii) a cross-entropy metaheuristic. Numerical studies show that the proposed greedy heuristic results in the optimal solution in more than 60% of the tested cases, while in almost 90% of the cases, the obtained solution is within 10% of the optimal solution. Both the approximated branch and bound and the cross-entropy methods achieved optimality in more than 80% of the tested cases; however, this came with a much higher computational cost. Copyright © 2016 John Wiley & Sons, Ltd.

Keywords: browsing time; expected path length; websites quality of service; websites analytics; web analytics

1. Introduction and motivation

Searching information content in websites is an integral part of internet services nowadays and is accepted as a major QoS criterion.[1,2] Users spend a great deal of time browsing *within* websites, searching for specific content items. A major quality measure of website browsing is the website service time, namely, the time it takes the user to reach a desirable content item once he or she reaches that website. While search engines, for example, Google, provide excellent mechanisms to find the content the user is looking for, in many cases, they target the homepage that is highly ranked by PageRank-like algorithms. In such cases, the internal search within the website requires additional browsing time, reflected by the browsing QoS while ignoring the external search. Accordingly, one of the major QoS challenges faced by website designers is how to better organize their sites, such that the content can be found effectively within the site. Let us note that this is a different challenge than organizing and optimizing the website such that its content is more visible and better oriented towards external search engines – a problem of importance by itself.

The challenge of saving man hours in online platforms is an appealing one to modern industrial engineers, just as much as the traditional industrial engineers have been focused on work standards and processes effectiveness in the past.[3,4] One measure of such QoS effort is the average number of steps (clicks) the users have to follow in the site until they reach their destination. Clearly, a website that enables the users to reach their desired pages effectively and quickly is considered to be of a better quality[5], providing a better user experience and is more likely to retain its users. This consideration causes many web designers to add more links among the website pages by *erroneously* thinking that a higher connectivity will reduce the website expected path length (EPL) and improve its QoS. In this paper, however, we show that such a policy is not necessarily effective and can result in an opposite situation.

To emphasize the importance of an effective website linking, let us refer to the Nielsen NetView Survey[6] in Figure 1. The survey shows the time spent on various internet sectors in the USA alone in 2010, measured in millions of hours/month. We focus *only* on the content-searching sectors that are candidates for the proposed method (i.e., Portals, Video/Movies, Search, Software Info, Entertainment and Classifieds that are marked in light gray), thus, sectors in which visitors should reach their desired content quickly and directly without moving from one page to another. The figure shows that during 2010, internet users in the USA alone spent 819 million hours a month online in these types of sites, both searching and consuming content items. Therefore, if one is able to save even 1% of this time[1] (assuming only a small portion of the time is actually spend on finding content, rather than consuming it), it implies a potential saving of more than 8.2 million man-hours each month. Even if the search time is only 0.01% of the browsing time,

Faculty of Engineering, Department of Industrial Engineering, Tel-Aviv University, Tel Aviv-Yafo, Israel
**Correspondence to: Irad Ben-Gal, Faculty of Engineering, Department of Industrial Engineering, Tel-Aviv University Israel.*
†E-mail: bengal@tau.ac.il

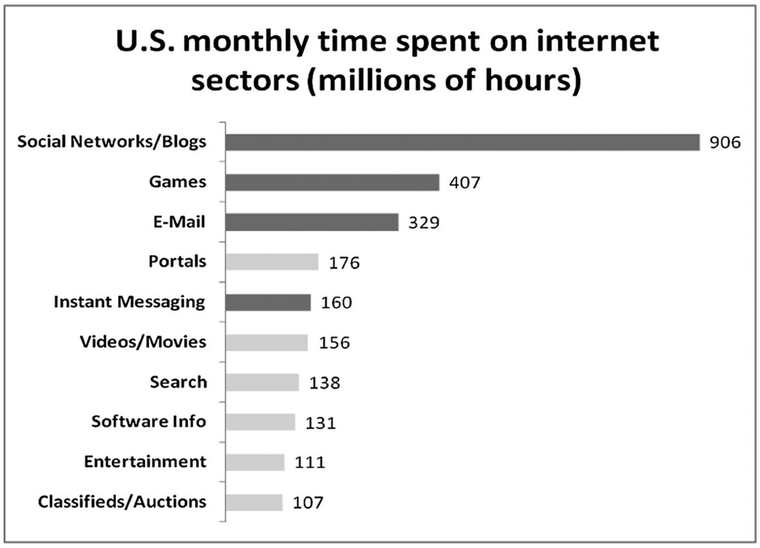

Figure 1. Nielsen NetView Survey, June 2010. US monthly time spent on most heavily used internet sectors (in millions of hours). Sectors depicted in dark gray are not considered relevant for this study and not included in our calculations

such a saving will result in almost 5 million man-hours per year according to the survey. Moreover, note that Nielsen's study was published before the mobile revolution that surely increases the time spent by users on content search in the internet. A survey that was published 4 years later in the Business Insider[7] shows a significant increase of the time spent by internet consumers in the USA that for some sectors quadruples in size, implying that the potential time saving of the proposed method is significantly higher.

Many analytical applications and papers, as indicated in the forthcoming literature review, address the effort of collecting and analyzing users' paths. Much less attention was devoted to what should be performed with such user information to improve the site structure – possibly in an adaptive and dynamic way. Because gathering user information is relatively straightforward nowadays, one should seek to exploit this data in order to increase the usability and QoS of the web site. In this paper, we show how information collected about the users can be used to redesign the website more efficiently with a better QoS metric.

There are several approaches to redesign the links' structure of a website automatically. The first is based on semantic analysis of the pages' contents, for example, matching contents by ontology-based text mining and artificial intelligence algorithms. The second and much simplest approach is based on the actual analysis of the users' traffic within the websites. In this paper, we follow the latter approach, which is particularly appealing as it makes use of information, which is readily available from web server logs,[8] without requiring the implementation of sophisticated semantic engines and artificial intelligence methods.[9] While many web designers believe that a website with many links benefits the users by allowing a higher connectivity, it is apparent that a structure that consists of many interlinks may result in a cluttered website that causes users to spend more time navigating until they reach their target. We focus on a method that redesigns a website only by removing links based on users' traffic information, aiming to minimize the EPL of the website without really affecting its content or its 'look and feel' user interface.

The study models the browsing behavior of users according to known conventions. For example, it assumes that users can be clustered by using a probability function over the destination URL (webpages) in the site.[10] Moreover, we follow previous studies that approximate the transition between pages by a state change of a discrete-time Markov chain.[10,11] The proportion of users in each cluster can be calculated while each cluster has its own transition probability function defined over the destination web pages. Often, it is assumed that users start at the website's home page and traverse the site pages by following a Markovian probability matrix until they reach their destination URL. The average number of pages traversed before arriving at the destination is used to estimate the EPL, which corresponds to the first passage time of the Markov Chain at the destination page. The focus of this study is to provide a method to redesign a website such that it allows faster access to desirable content for different users. We measure the EPL on a Markov model presented by Zukerman[12] as a performance measure for the accessibility of the site and introduce some heuristic methods to minimize the EPL by removing some of its links.

For illustration purpose of the proposed approach, consider a schematic example of a website of a large medical clinic. The site contains several pages related to the different services that a potential patient can obtain, such as, scheduling doctor appointments, listing of medical services, viewing lab reports, etc. While such sites are often organized hierarchically, they may contain links between webpages of the different services. Moreover, site builders will often favor adding many such links, thinking that this will result in a well-connected, easy-to-navigate site with a smaller EPL figure. In many cases, however, this is a misconception – adding links may result in a more complex site that is harder to navigate, resulting in a larger EPL for a user to reach his or her destination. Accordingly, the question here is whether to remove some of the links so that the average time to reach a specific content is minimized.

In the illustrative example shown in Figure 2, two paths exist to reach the *Lab Reports* URL: (i) browsing through the *Lab Test* URL (under the assumption that one could view the report from the page of their appointment); or (ii) reaching there straight from the

[1]*Several blogs (e.g., http://www.go-gulf.com/blog/online-time/) claim that search time can reach up to 21% of the total time people spend online.*

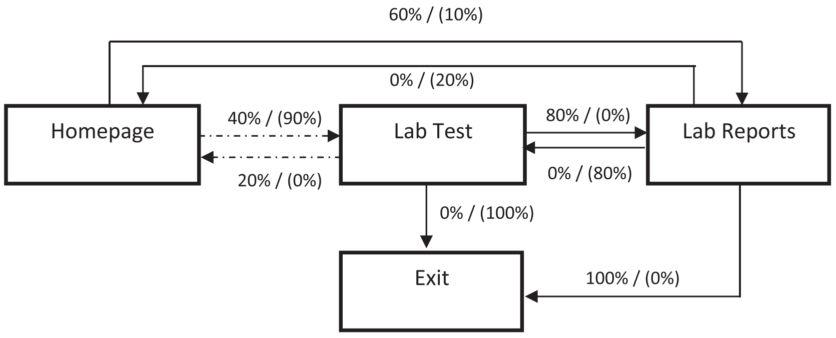

Figure 2. Schematic view of a transition matrix and link deletion in a medical website. Two transitions metrics are given as follows: (i) percentage showing the transition probability for people aiming to reach the Lab Reports URL (shown without parentheses); (ii) percentages (in parentheses) that show the transition probability for people aiming to reach the Lab Test URL. Deleting the dashed line from Homepage to Lab Test results in shorter expected path length to Lab Reports

Homepage URL. Transition probabilities for the people aiming to reach the *Lab Reports* URL are shown without parentheses. Note that in this example, if most of the users are aiming to reach the *Lab Reports* URL, it may be better to remove the link from the home page to *Lab Test* URL, forcing all users to surf the site via the *Lab Reports page*. By using the server log data, it is possible to calculate the first passage time from the homepage to each target page. In the example, the first passage time for users aiming towards the *Lab Reports* URL is 1.52 steps (clicks), while the first passage time for users aiming towards the *Lab Test* URL is 1.12 steps. By deleting the link from the homepage to the *Lab Test* URL, the former passage time is reduced to 1.0, while the latter passage time is increased to 2.5. Yet, because most of the users seek the *Lab Test* URL, the overall EPL is shortened. These opportunities to improve the site QoS exist in more complex websites and are considered in the paper.

In this paper, we propose and analyze several algorithms for redesigning a website by deleting links in order to decrease the site EPL. The first algorithm is a greedy deletion (GD) heuristic that deletes, at each iteration, the link having the maximum marginal contribution to the EPL. The second algorithm is an approximated branch and bound (AB&B) heuristic. This heuristic employs a branch and bound search strategy over the entire solution space, namely, it approximates an EPL bound at each node of the search tree. The third algorithm is based on the cross-entropy (CE) metaheuristic that was originally introduced by Rubinstein[13]. This evolutionary metaheuristic is based on a parameterized random search mechanism that generates random solutions. The parameters of the search mechanism are gradually updated by minimizing an entropy function, seeking to increase the likelihood of potentially 'better' solutions to be considered in the next search iteration.

Results from the aforementioned heuristics show that both the GD and the CE heuristic perform well and relatively fast, while the branch and bound method is clearly dominated by them. The CE heuristic, for example, achieved the optimal structure solution in 87% of the sample set for websites with 24 pages at an average time of less than 1 min (on a 2-cores 16GB computer). The GD heuristic reached the optimal solution in 64% of the same sample set, obtaining these solutions at an average time of less than 1 s on the same machine. Moreover, most of the suboptimal solutions of the GD heuristic are found to be within a 10% range of the optimal solution. Such a performance indicates that while the CE algorithm can be used for offline design of sites, the GD algorithm is fast enough to be implemented online, enabling a personalized browsing experience for each type of user depending on their browsing statistics.

The rest of this paper is organized as follows: Section 2 presents some relevant literature papers. Section 3 introduces the minimum EPL (MEPL) problem, including a formal mathematical model. Section 4 presents the proposed solution methods for the MEPL problem. In Section 5, the effectiveness of the proposed heuristics is benchmarked against the optimal solution that is obtained by a complete enumeration of the solution space for some small instances of the problem. Section 6 concludes the work and indicates some future research directions.

2. Related work

More than a decade ago that Perkowitz and Etzioni[14,15] described several challenges faced by website designers that seem to be relevant also today: (i) Different visitors may have dissimilar goals; (ii) The same visitor may seek different information at various times; (iii) Many sites outgrow their original design, accumulating links and pages in unlikely places; and (iv) The designers' a priori expectations regarding the usage of the site may be violated. One of their proposed approaches to address these challenges was to design an automatically adaptive website, that is, websites that automatically improve their organization and presentation by learning from users browsing patterns. Perkowitz and Etzioni further noted that websites may either be adapted to the individual user or to a community of users as a whole, as proposed in this work that targets a cluster of users.

Some of the challenges mentioned in Perkowitz and Etzioni[14,15] are addressed today in many commercial websites. For example, the Amazon.com site presents personalized commerce contents to users according to their past purchases, aiming to improve the user experience, shorten the transaction time, and increase convergence. iGoogle provides a customizable platform that allows each user to define its preferences on the structure and the presented content. This allows each user to design a personal homepage browser with content accessories related to his or her own interests. For example, an academic researcher may seek a quick access to the 'Google Scholar' search engine, while a gamer may look for constant RSS feeds related to new game releases to be shown in his homepage.

A preliminary condition for websites' personalization is the ability to learn the user preferences. And indeed, various data-mining and machine learning models were suggested in recent years to learn such preferences and use these automatically to build adaptive sites, among other applications. Web-mining nowadays is used extensively to profile users who access particular URLs based on demographic/behavioral/personal information that is available from various resources, including cookies, smartphones, location sensors, and various apps. Classification based on access logs is used to uncover segmentation information such as the IP addresses of most of the users who often access a specific URL, as well as their profiling. For example, a common association rule based on the aforementioned information can be of the following format '50% of the users who placed an online order for product X of company Y belong to a segment of 20–25 years olds living in a large city on the west coast of the USA'. *Classification* methods are now used to map users into predefined classes (for example by collaborative filtering), and a different website structure can then be presented to each class. For example, if a user is classified as a fan of certain games or movie types, he or she will be automatically directed to relevant sites. Segmentation and web-mining enable to support dynamic modifications of websites for new or revisited clients that follow several of the approaches proposed by Etzioni[14] and others. Roughly indicating, many of the aforementioned models can be categorized into two classes: (i) *Web-mining models*, such as association rules, sequential pattern discovery, clustering, and classification (e.g., [16,17]); and (ii) *Predictive Statistical models*, such as linear models, neural networks, Bayesian networks, and Markov chains (e.g., [11,18]), that are used in this work.

One example for a machine-learning application that tracks browsing behavior and applies it to websites' link-design was proposed in Mobasher *et al.*[16] The authors used association rules, collaborative filtering, and web-mining techniques to find webpages that are connected implicitly by users' browsing behavior. They found significant correlated accesses to various URLs and indicated that such correlations can be used to generate new links between the correlated websites. For example, if 60% of the users who accessed the page www.amazon.com/Under-Dome-Novel-Stephen-King/dp/1439149038/ later also accessed the page www.amazon.com/Hunger-Games-Suzanne-Collins/dp/0439023521/, then the former page may contain a referral link to the latter. The authors argue that the advantage of using web-mining techniques is that the search space can be more effectively pruned, partly because of the fact that most sites are organized in a hierarchical fashion. For example, if the URL */company/products* has few hits (entries), it is safe to assume that the lower-level page */company/products/products1.html* will have even fewer hits and, therefore, could be pruned. Thus, revealing the association rules of the site may help to organize it more efficiently, as we aim to do in this work by proposing a different application.

Other examples for applications based on personalized browsing behavior can be indicated. Jiang and Tuzhilin[19] presented a segmentation approach that is not based on computed statistics, but rather on directly combining transactional data, such as web browsing and purchasing activities of several users. Baraglia and Silvestri,[20] Saad,[21] and Jalali *et al.*[22] discussed a Web Personalization (WP) application based on web usage mining that customizes the content of a web site with respect to the users' needs. They tracked the users' browsing behavior and proposed a mechanism to predict the next content of interest for a user. Saad and Kruschwitz[23] further illustrated the use of web usage mining for an adaptive website. Goyal *et al.*[24] proposed an online hotlink assignment algorithm for designing adaptive websites, with a desired goal to reach certain contents on the site within a minimal number of clicks. The authors looked at the website as a directed graph and allowed an addition of up to *k* hotlinks on each page. Such an approach is now adopted by paid-content companies such as Taboola and Outbrain. Chen and Ryu[25] suggested a model in which the website designer adds a certain amount of links to the structure of a website to provide a better navigation. They set a threshold value for the number of paths a user will agree to pass through before churning from the site. The authors assumed that by adding links, up to a threshold number, the user will be directed to the desired content faster. Similarly, Jadhav and Bhale[26] proposed a mathematical programming model for adding or enhancing website links to improve the website structure while minimizing the required website's changes (assuming that users' backtracking is undesired). In this paper, we follow an opposite approach, and instead of adding links, we consider the deletion of up to *k* links among webpages. The proposed approach is advantageous because it is simple, does not require a change in the website design, and often reduces the browsing EPL.

Popular models that are used to estimate the EPL of a website, and also used in this paper, are Markov models that are generated by historical browsing data. These models can be used to predict the next link 'click' given an observed browsing path, while estimating the first passage time from one page to another. For example, Bestavros[27] used a Markov model to calculate the probability that a user will seek a particular document based on his browsing behavior. Zukerman *et al.*[28] compared the predictive performance of different Markov models when forecasting a user's next request. Let us note that all these methods can be used to modify the hierarchy or the layout of a website once the next URL is predicted with high enough probability. We follow these basic ideas in the proposed modeling.

Finally, let us note that in the past few years, the development of well-structured websites has become an even bigger challenge, probably because of the popularity of e-commerce sites, in which a poor link structure and content visibility often result in a direct loss of revenue. Multiple researchers suggested new heuristics to solve the Website Structure Optimization (WSO) problem, as it is called now when addressing not only the linking but also structure and content optimization. For example, Yin and Guo[29] suggested an enhanced Tabu Search procedure to solve the WSO problem. They showed how website owners can obtain a feasible high-quality solution, although often not optimal. Paul and Kumar,[30] Jadhav and Bhale[26] both suggested procedures for adding or repairing existing website links to improve user navigation through a website. For further reading on the extended WSO problem, the reader is referred to Singh and Kaur[31] that provide a survey on the topic.

3. Problem formulation and notation

The link structure of a website is modeled by a sparse directed graph. Each node is related to a page, and each arc is related to a link between an ordered pair of pages. The transition probability between pages is assumed positive if and only if the pages are directly linked. The transition probability information is readily available from the website's log data. The designer of the website may remove links between pages. By doing so, he or she sets the transition probability between the corresponding pages to zero, while proportionally increasing the transition probability to other pages (normalizing the rest by the sum of probabilities). For example, the link structure of a website with four pages may resemble the one in Figure 3, having the corresponding transition probabilities:

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 0.25 & 0.75 \\ 0 & 0 & 0.5 & 0.5 \\ 0.65 & 0.05 & 0 & 0.3 \\ 0.1 & 0.2 & 0.7 & 0 \end{pmatrix}.$$

In this example, a user moves from page 1 (the homepage) to page 3 with a probability of 0.25 and to page 4 with a probability of 0.75. There is no direct route from the homepage to page 2. Therefore, a user who wishes to reach page 2 from the homepage will go first to either page 3 or 4, and from there, his or her destination can be reached. The designer might aim at decreasing the EPL of the users in a given website by solving the MEPL problem. The MEPL problem is defined as follows: given an integer number D, delete up to D links in the web site so as to minimize the EPL.

Following the example, let us present the notation that will be used in the proposed mathematical model:

Parameters:

n	Number of pages in the site. Page 1 represents the homepage
L	Total number of links in the site (initially)
D	Number of links for deletion
p_{ijt}	The transition probability from page $i = 1, \ldots, n$ to page $j = 1, \ldots, n$ of users with final destination t. We denote the $n \times n$ stochastic matrix as **P** and refer to it as the Transition Probability Matrix. There are no self-loops, so $p_{iit} = 0$.
w_t	The proportion of visitors with destination page $t = 1, \ldots, n$. We denote the vector of proportions by **w** and refer to it as the *content proportion vector*.

Decision variables:

x_{ij}	A binary variable that represents whether the link between page i, and j is kept. These variables are defined only for existing links that are traversed by some users, that is, $\sum_t p_{ijt} > 0$.

Auxiliary variables:

μ_{ijt}	The mean first passage time at page j, for users starting at page i with final destination t.
v_{it}	Normalizing variable representing the change of transition probabilities following a deletion of links at page i, such that the total transition probability out of page i remains one.

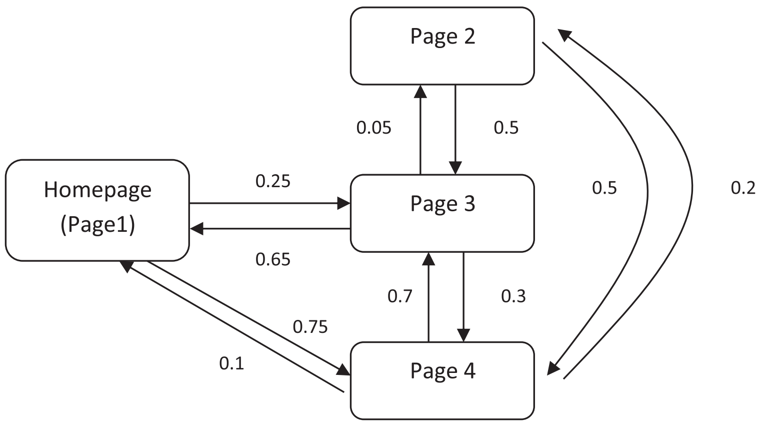

Figure 3. Example of a website with four pages and the linking structure, represented by a Markov chain

Assumptions:

- The model focuses on deleting a webpage's internal links in order to reduce the EPL. Although the objective could be achieved by adding new links, such a solution requires a different analysis approach and left for future research.
- The model does not address additional issues related to webpage optimization, such as the webpage rank scoring, adaptability to search engines, ease-of-use, and other user experience objectives.
- The model follows previous studies that approximate the transition between pages by a state change of a discrete-time Markov chain.[10,11] In particular, we use a discrete-time Markov chain, which focuses on analyzing the expected travel steps before reaching the target page. We further assume that the average number of traveling steps is proportional to the average search time.
- Upon link deletion, the new transition probabilities of the new network are normalized such that the probability associated with the deleted link is proportionally redistributed to the other pages.
- The browsing behavior of the users is stationary, and the transition between pages can be approximated by a state change of a discrete-time Markov chain.[10,11]

Following the above, we are mostly interested in $\mu_{1,j,j}$, the first passage time at page *j* of users starting at the homepage with page *j* as their final destination. μ_{ijt} is evaluated by solving the following linear system of equations:

$$\begin{aligned} \mu_{ijt} &= 1+\sum_{k=1}^{n} p_{ikt}\mu_{kjt} \quad \forall i, j\neq i \\ \mu_{i,i,t} &= 0 \quad \forall i \end{aligned} \tag{1}$$

The MEPL problem is then defined mathematically as follows:

$$\begin{aligned} &Min\ \textstyle\sum_j \mu_{1jj} w_j \\ &s.t. \end{aligned} \tag{2}$$

$$\mu_{ijt} = 1+\textstyle\sum_{k:p_{ik}>0} v_{it}x_{ij}p_{ikt}\mu_{kjt}\ \forall\ i,\ j: \neq i, t \tag{3}$$

$$\textstyle\sum_i \sum_{j:\sum_t p_{ijt}>0} x_{ij} \geq L - D \tag{4}$$

$$\textstyle\sum_{k:p_{ikt}>0} v_{it}x_{ik}p_{ikt} = 1 \quad \forall i\ , t \tag{5}$$

$$x_{ij}\in\{0,1\} \qquad \forall i, j: \textstyle\sum_t p_{ijt} > 0 \tag{6}$$

$$\mu_{ijt} \geq 0\ \forall i, j, t \tag{7}$$

$$v_{it} \geq 0 \quad \forall i, t \tag{8}$$

The objective function (2) minimizes the EPL. Constraint (3) calculates the first passage time based on the modified transition probabilities following the deletion of links. Constraint (4) limits the number of allowed deletions. Constraint (5) normalizes the transition probabilities following a deletion. Constraints (6), (7), and (8) are integrality and non-negativity constraints.

The previous non-linear and non-convex mathematical model cannot be solved by using standard branch and bound techniques that are supported by commercial solvers. Accordingly, three algorithms are proposed and evaluated in the next sections.

4. Proposed algorithms

Three different heuristics are proposed based on the considered MEPL problem. The algorithms are as follows: (i) a GD heuristic; (ii) an AB&B heuristic; and (iii) a CE based heuristic. In this section, we present these algorithms, while in the next section we show the results of a numerical experiment that compares the performances of these algorithms against the optimal solution that was obtained by an exhaustive search (ES). The next subsections outline the three heuristics, providing both a pseudo code and a running example on each one of them.

4.1. Greedy deletion heuristic

This algorithm greedily deletes arcs one at a time. In each iteration, the link whose removal contributes the most to reducing the EPL is removed. The process is terminated once no link deletion contributes positive value to the objective function, or after *D*, links were removed. In order to calculate the effect of a link removal, its probability is temporarily assigned to other links of the same page, as described in Section 3. The algorithm is formally described in Pseudo Code Listing 1.

```
Greedy Deletion Algorithm for MEPL
Input: Transition matrix P, maximal number of links to delete $D$
Calculate transition matrix $EPL$ and set $TempEPL = EPL$
Set $MinEPL = TempEPL$
While $D \geq 0$ do
        Set TempP = P
        BreakFlag = True
        For $i = 1, \dots, n$
            For $j = 1, \dots, n$
            If $P[i,j]$ =0 then
                    TempP[i,j] = 0
                    For $z = 1, \dots, n$
                        TempP [i,z] = $TempP[i,z]/(1 - P[i,j])$
                    Calculate $EPL$ for TempP
                    If $EPL < MinEPL$ then
                        .
                        disc_i = i, disc_j = j
                        BreakFlag = False
        If BreakFlag then Break (while loop)
        $D = D - 1$
        For $z = 1, \dots, n$
            P [disc_i,z] = P[disc_i ,z]/(1 - P[disc_i , disc_j])
        P[disc_i , disc_j] = 0

Return P
```

Pseudo code listing 1.

Illustrative Example As an illustrative example, consider the following 5×5 matrix and allow for the deletion of up to three links. Accordingly, the following notations and parameters are used:

$$\mathbf{P} = \begin{pmatrix} 0 & 0.3297 & 0.1087 & 0.2286 & 0.3331 \\ 0.1729 & 0 & 0.2386 & 0.1776 & 0.4109 \\ 0.2529 & 0.1968 & 0 & 0.3280 & 0.2224 \\ 0.2937 & 0.1641 & 0.3078 & 0 & 0.2343 \\ 0.0485 & 0.2670 & 0.4750 & 0.2096 & 0 \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} 0.074 \\ 0.051 \\ 0.7 \\ 0.057 \\ 0.116 \end{pmatrix}$$

Thus, **P** is the transition matrix, and **w** is the content proportion vector, and $D = 3$.

The original EPL matrix is calculated. The calculation is obtained by solving the linear equation of the first passage time between each pair of pages, as seen in Equation (1). This results in the matrix $\boldsymbol{\mu}$. The EPL can now be calculated by multiplying the matrix $\boldsymbol{\mu}$ by the content proportion vector **w**, resulting in 3.4858 steps for this example.

The next step is to delete the first available link, which is link (1,2) with an initial probability $p_{12} = 0.3297$. While p_{12} is set to zero, all the other probabilities in the row are normalized by their cumulative sum. That is,

$$p_{1,j} \leftarrow \frac{p_{1j}}{1 - p_{12}} \quad for\, j = 3, 4, 5.$$

Thus, a user positioned at page 1 can no longer move to page 2 directly and has to move to pages 3, 4, or 5 (assuming page 1 is not his or her goal), before reaching page 2. Recalculating the EPL based on the new transition matrix results in a new EPL:

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 0.1621 & 0.3410 & 0.4969 \\ 0.1729 & 0 & 0.2386 & 0.1776 & 0.4109 \\ 0.2529 & 0.1968 & 0 & 0.3280 & 0.2224 \\ 0.2937 & 0.1641 & 0.3078 & 0 & 0.2343 \\ 0.0485 & 0.2670 & 0.4750 & 0.2096 & 0 \end{pmatrix} \qquad \text{EPL} = 3.2534.$$

As seen, a lower EPL is obtained by deleting the link (1,2). Applying the same method, we now return to the initial matrix and delete the next available link (1,3). This deletion results in the following transition matrix and EPL:

$$P = \begin{pmatrix} 0 & 0.3699 & 0 & 0.2565 & 0.3737 \\ 0.1729 & 0 & 0.2386 & 0.1776 & 0.4109 \\ 0.2529 & 0.1968 & 0 & 0.3280 & 0.2224 \\ 0.2937 & 0.1641 & 0.3078 & 0 & 0.2343 \\ 0.0485 & 0.2670 & 0.4750 & 0.2096 & 0 \end{pmatrix} \qquad \text{EPL} = 3.7873.$$

Note that deleting link (1,3) leads to a larger EPL than the one obtained by deleting link (1,2) and even larger than the initial EPL. In the same manner, the proposed procedure evaluates the deletion of all available links. It turns out that the maximal reduction in the EPL is obtained by deleting link (5,2). Having this link removed, one obtains the following transition matrix and its corresponding lowest EPL for the first deletion:

$$\mathbf{P} = \begin{pmatrix} 0 & 0.3297 & 0.1087 & 0.2286 & 0.3331 \\ 0.1729 & 0 & 0.2386 & 0.1776 & 0.4109 \\ 0.2529 & 0.1968 & 0 & 0.3280 & 0.2224 \\ 0.2937 & 0.1641 & 0.3078 & 0 & 0.2343 \\ 0.0662 & 0 & 0.6479 & 0.2859 & 0 \end{pmatrix} \qquad \text{EPL} = 3.1957.$$

The procedure now continues for up to $D = 3$ iterations (the number of permitted deletions) and eventually reaches the best solution by deleting links – (5,2), (5,4), and (4,1). The following transition matrix and its corresponding EPL are presented as follows:

$$\mathbf{P} = \begin{pmatrix} 0 & 0.3297 & 0.1087 & 0.2286 & 0.3331 \\ 0.1729 & 0 & 0.2386 & 0.1776 & 0.4109 \\ 0.2529 & 0.1968 & 0 & 0.3280 & 0.2224 \\ 0 & 0.2324 & 0.4359 & 0 & 0.3318 \\ 0.0926 & 0 & 0.9074 & 0 & 0 \end{pmatrix} \qquad \text{EPL} = 2.7412.$$

Thus, in this example, one can obtain a reduction in the EPL of more than 20%. Note that the complexity of this procedure (and the following two procedures) is a constant function of the inventions of the transition matrix. Thus, the complexity of the greedy heuristic is $O(L \cdot MI(L) \cdot D)$, where $MI(L)$ represents the complexity of the inverse operation of a sparse matrix with L non-zeros. For example, if one uses the Gauss–Jordan elimination $MI(L) = O(L^3)$, the resulting complexity of the greedy heuristics is $O(L^4 D)$.

4.2. *Approximated branch and bound heuristic*

In this section, we present a heuristic that is based on an AB&B framework. The proposed method is similar to the conventional Branch and Bound strategy, except that the lower bounds are conjectured based on heuristic considerations and are not necessarily valid. Such an approximated lower bound might lead to the erroneous pruning of branches in the search tree that might contain the optimal solution. Nevertheless, as demonstrated later, if the lower bounds are 'guessed' plausibly, this approach may lead to a successful solution method.

Let us present some notations that are used for a more formal description of the heuristic.

LINK	– the list of links in the website ordered in some arbitrary manner. The individual links are denoted as LINK(1), LINK(2),…, LINK(L)
TREE	– list of 'active' (unprocessed) nodes in the search tree.

Each node consists of a tuple (*DEL*, *CUR*, *EPL*), where

DEL	– Set of links that must be deleted in any offspring of this node
CUR	– the index (in LINK) of the previously inspected link (0 if no link was inspected yet).
EPL	– The expected path length of the website after removing the links in DEL.

Our approximated bound is based on the assumption that the reduction rate of the EPL, because of the links that where deleted during the branch and bound process, also represents the expected rate of EPL reduction for the remaining links to be deleted. We denote the initial EPL by EPL_0, and thus, the total improvement up to a given branch and bound node is given by $EPL_0 - NODE.EPL$, which represents a reduction rate of

$$\frac{(EPL_0 - NODE.EPL)}{|NODE.D, E, L|}$$

per link, where $|NODE.DEL|$ is the number of deleted links at that branch and bound node. Assuming the same average reduction rate remains effective for all the D links to be removed next, we obtain the following approximated lower bound, where

$$ALB = EPL_0 - D \cdot \frac{(EPL_0 - NODE.EPL)}{|NODE.D, E, L|}.$$

Because the branching criteria favors the most 'promising' links for first removal, the above lower bound is likely to be valid. For the root node, where $|NODE.DEL| = 0$, we set $ALB = 0$, which is surely a valid (yet a very loose) lower bound.

The AB&B algorithm is presented in Pseudo Code Listing 2.

```
Approximated Branch and Bound Algorithm for MEPL

Input: Transition matrix $P$, maximal number of links to delete D
Create the list $LINK$ (based on the non-zero elements in $P$).
Calculate EPL for the initial website design and store it in $EPL_0$.
Insert into TREE the node $(\varnothing, 0, EPL_0)$
Set $INCUMBENT = EPL_0$
While $TREE$ not empty

        Remove the $NODE$ with the smallest NODE.EPL from array $TREE$
        Calculate the greedy solution where all the links in $NODE.DEL$ are forced to be deleted and only links with an index greater than $NODE.CUR$ are considered for further deletion.
        Store the value of the best solution in $UB$

        IF $UB < INCUMBENT$ set $INCUMBENT = UB$
        Set approximated lower bound
            $ALB = \begin{cases} EPL_0 - D\cdot\frac{(EPL_0 - NODE.EPL)}{|NODE.D,E,L|} & , NODE.DEL \neq \varnothing \\ 0 & , NODE.DEL = \varnothing \end{cases}$

        IF $(ALB < INCUMBENT)$ and $(|NODE.DEL| < D)$

            For $i = NODE.CUR + 1$ to $|LINK|$
                Calculate EPL for $NODE.DEL \cup \{LINK(i)\}$ and store in $X$
                Insert into $TREE$ a node $(NODE.DEL \cup \{LINK(i)\}, i, X)$

Return INCUMBENT solution
```

Pseudo code listing 2.

Illustrative Example Consider a website with a transition matrix **P** and five existing links with a maximum of two disconnections permitted. Assume that the original design's EPL is 2, that is, $EPL_0 = 2$. Initially, the node $(\varnothing, 0, 2)$ is inserted into the array *TREE*, and we set *INCUMBENT* = 2.

The node with the corresponding smallest EPL is removed from the array TREE. Because initially there is only one node, we remove it, and the greedy solution, as described in Section 4.1 with no prior-deletions, is calculated (because $NODE.DEL = \varnothing$). Let us assume that the greedy solution for this node is 1.5 steps. This value is set as the current *UB*, and because it is better than the incumbent solution, it is also set as the *INCUMBENT*.

Because so far no links were actually deleted, we set $ALB = 0$.

Because $(ALB < INCUMBENT)$ and $(|NODE.DEL| < D)$, we add a node into the *TREE* with all possible deletions of one link with an index greater than *CUR*. For each such node, we calculate the *EPL* assuming only the links in *NODE.DEL* are deleted.

We now return to the beginning of the algorithm and remove the next node with the smallest EPL from array *TREE* for processing. As can be seen in Figure 4 later, this is node $(\{3\}, 3, 1.7)$. We calculate the greedy solution of this node and find its value is 1.5. This is the UB and as it is equal to the incumbent solution, which does not change. We calculate $ALB = 1.4$. Because the *ALB* is lower than the incumbent solution, we continue with this branch.

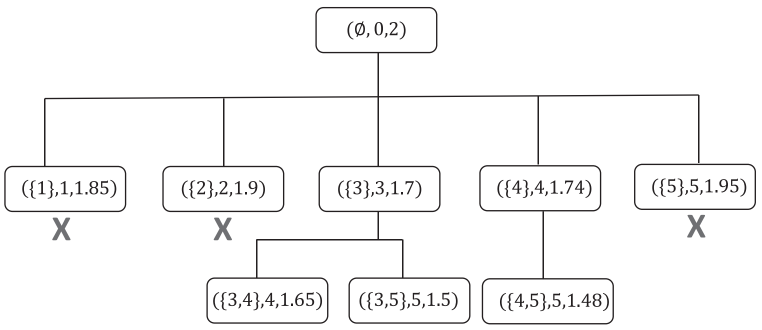

Figure 4. Approximated branch and bound example for a site with five links

Bounding is performed when the *ALB* is greater than the incumbent solution. For example, node ({1}, 1, 1.85)'s greedy solution resulted in 1.72 steps. This is the UB for this solution. This is not an improvement of the incumbent solution, which was calculated as 1.5. The *ALB* is then calculated as 1.7. Because the *ALB* is greater than the incumbent solution, the branch is bound.

We continue for each un-reviewed node and stop once all the tree nodes have been processed. Figure 4 shows the order in which the nodes enter the algorithm. Eventually, we bounded three nodes for having an *ALB* greater than the incumbent solution (example omitted).

As can be seen, node ({3, 5}, 5, 1.5) reached the EPL obtained by the initial matrix. However, node ({4, 5}, 5, 1.48) reached a lower EPL, and because no other deletions are allowed (two deletions already exist in *NODE.DEL*), this is the final obtained solution.

4.3. The cross-entropy heuristic

The CE method[12] generates a random data sample of solutions, finds the best solutions, refines the data generating mechanism accordingly, and produces a 'better' sample in the next iteration. The process continues until eventually converging to a single solution. A *solution* in our problem definition is a set of *D* deleted links.

The notations used for this heuristic, in addition to the original notations given earlier, are as follows:

$\boldsymbol{N}$	– the number of solutions generated in each generation
$\boldsymbol{T}$	– the top percentile of observations that are used as an *Elite* set. For example, $\boldsymbol{T} = 10\,\%$ implies the algorithm only uses the top 10% best solutions to refine the random mechanism for the next iteration.
$\boldsymbol{P_i}$	– defines the probability vector in step *i*. This vector represents our chance of choosing a certain link to be disconnected. $\boldsymbol{P}_0$ is the initial probability vector.
$\boldsymbol{S}$	– The probability breaking point. When *D* links have probability equal or greater than $\boldsymbol{S}$ in the $\boldsymbol{P_i}$ vector, the process stops, and this vector is returned as the final solution.
α	– Represents the smoothing coefficient used to create the next generation probability vector.

At the beginning of the process, $\boldsymbol{P}_0$ is defined as a uniformly distributed vector of length $\boldsymbol{L}$. Because eventually this probability vector should equal the number of disconnections allowed (and not 1), each link is given the probability of $\frac{\boldsymbol{D}}{\boldsymbol{L}}$.

Using the probability vector, *N* solutions are randomly generated, where each solution represents a website with *D* links disconnected. Out of those *N* solutions, only the top *T* solutions are observed. A new probability vector is now defined, which equals the percentage of times each link got disconnected in the top solutions (we use the following notation to represent this new vector: $\boldsymbol{AP_i}$).

Because certain links may not have been disconnected in the top *T* solutions, their new probability will be set to zero, meaning that they will not be chosen in the next iteration (or ever). To prevent this from happening, the coefficient α is used to smooth the new probability vector with the former generation probability vector:

$$\boldsymbol{P_i} = \boldsymbol{AP_i}\cdot \alpha + \boldsymbol{P_{i-1}}\cdot(1-\alpha)\ .$$

The new vector $\boldsymbol{P_i}$ will be our probability vector for the next iteration. The process finishes when the probability vector consists of *D* links with probability equal or greater than *S*.

The entire process is outlined in Figure 5.

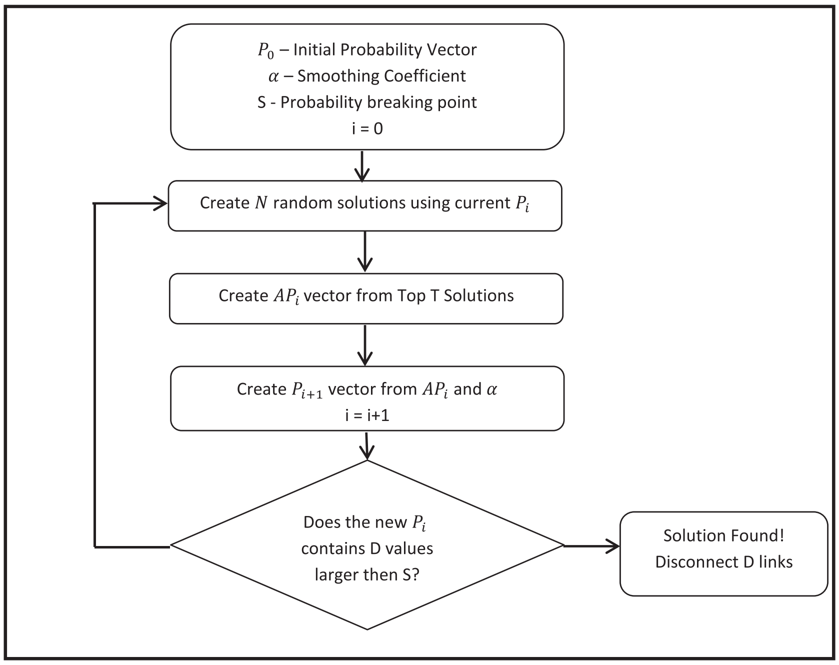
P_0 – Initial Probability Vector
α – Smoothing Coefficient
S - Probability breaking point
i = 0
Create N random solutions using current P_i
Create AP_i vector from Top T Solutions
Create P_{i+1} vector from AP_i and α
i = i+1
Does the new P_i contains D values larger then S?
Solution Found! Disconnect D links

Figure 5. Outline of the solution evaluation process for the cross-entropy heuristic

Illustrative Example Let us look at a simple example using a site, which consists of $L=10$ links, $N=100$, $T=10\,\%$, $D=3$, $\alpha=0.6$, and $S=0.96$.

The initial $\boldsymbol{P}_0$ is a uniformly divided vector of 10 elements with a value of 0.3 for each one. The algorithm uses $\boldsymbol{P}_0$ to generate N solutions, each with D links disconnected. It then observes the top 10% ($T=0.1$) of the solutions and finds the number of times each link appears. Using this observation it calculates the $\boldsymbol{AP_i}$ and, eventually, the $\boldsymbol{P_{i+1}}$:

$$P_i=\begin{bmatrix}0.3\\0.3\\0.3\\0.3\\0.3\\0.3\\0.3\\0.3\\0.3\\0.3\end{bmatrix}\rightarrow\ \#=\begin{bmatrix}3\\6\\5\\3\\8\\2\\2\\1\\0\\0\end{bmatrix}\rightarrow AP_i=\begin{bmatrix}0.3\\0.6\\0.5\\0.3\\0.8\\0.2\\0.2\\0.1\\0\\0\end{bmatrix}\rightarrow P_{i+1}\ (\alpha=0.6)=\begin{bmatrix}0.3\\0.48\\0.42\\0.3\\0.6\\0.24\\0.24\\0.18\\0.12\\0.12\end{bmatrix}.$$

The process continues until reaching D links with probability equal or greater than S.

The complexity of each iteration of the CE method is $O(N\cdot MI(L))$. Again, using the Gauss–Jordan elimination method, the complexity is $O(N\cdot L^3)$.

5. Numerical experiments and results

The goal of the numerical experiment is to evaluate the proposed link-deletion methods and find the heuristic that results in the lowest EPL in various experimental conditions. In order to benchmark the proposed algorithms, a set of problem instances were generated. Each instance consisted of a directed graph that represents the link structure of the website, its transition probability matrices, and the content proportion vector (describing the distribution of the target pages of the users).

Scale Free transition matrices were generated randomly. Websites, like many other networks in nature, are typically structured as Scale-Free Networks,[32] representing a sparse network with a few nodes of high degree (commonly referred to as '*hubs*') and many nodes with a low degree. The procedure used to generate a scale-free network (or Matrix) was based on the *linear preferential attachment*[32] principle, as follows:

1 A network is constructed using a set of m_0 vertices.
2 New vertices are added up to the dimension of the matrix required.
3 Each new vertex is connected with no more than m links to the previously added vertices.
4 The new edges are chosen using a *linear preferential attachment* rule, that is, the probability of the new vertex i being connected to an existing vertex j is proportional to the degree of j. This rule is also known as 'the rich get richer' paradigm and follows the power-law principle.

For the purpose of simplicity, we assume that the transition matrices of all the users were identical and independent of their target URL. The diagonal of the transition probability matrices were kept to zero in order to rule out links/movements from a page to itself (i.e., the *Refresh* button is not considered a movement). Using two m links values, either $m=2$ or $m=3$, we generated 120 different transition probability matrices and website structures: 40 of them contained 8 pages, 40 contained 16 pages, and 40 contained 24 pages (for validation purpose several experiments were executed on larger websites with 50, 100, and 200 pages that are discussed in Section 6). These website sizes represent of small and medium organizations that form the majority of the websites in the WWW. Note that the used network-generation procedure describes a case where new added webpages are most probably connected to the homepage, resulting in a scale free network where hubs have degrees higher than m.

The *content proportion vector* was generated with three different methods:

1 Using a randomly generated vector – initially, set $p_{ij}=U(0,1)$ for all existing links (i,j) and then normalize the probabilities by dividing the elements of each row i by the sum of probabilities $\sum_j p_{ij}$.
2 Using a uniform distribution – the transition probability from page i to each linked page j is $p_{ij}=\frac{1}{k}$, where k is the number of links at page i.
3 Using a 70/30 vector – one of the pages is randomly selected (t'). Then, $w_{t'}=0.7$. Other pages $t\neq t'$ were then randomly assigned a portion of the additional 0.3.

For each of the 120 matrices described earlier, we generated a content proportion vector using each one of these three methods. Each of these 360 instances were tested with $D=4$ and $D=3$. In total, 720 instances were created for each website in each of the three sizes.

We ran all three heuristics, namely, GD, AB&B, and CE, on each instance. In order to compare the obtained EPL results with the optimal solution, we used ES over all the $\binom{L}{D}$ possible solutions. The ES results are denoted by OPT. The three algorithms and the ES were implemented in MathWorks MATLAB version R2009b using Microsoft Windows 7. The experiment was executed on an Intel i7 with 8GB.

Two metrics of interest are observed: (i) avg. run time – the average time necessary to obtain a result (in seconds); (ii) the optimality gap – measuring how close a solution is to the optimal solution. The optimality gap is calculated for each heuristic as shown in Table I below:

$$Optimality\ Gap = \frac{Heuristic\ Solution - OPT}{OPT}.$$

Note that *OPT* is the true minimal EPL of the website obtained by ES, and *Heuristic Solution* is the actual EPL obtained by one of the faster heuristic solutions.

Table I summarizes the results of the experimental runs for the three groups according to the dimension (number of pages) of the website, as shown in the first column. The second column describes the solution heuristic that was used to modify the links. The third column shows the average running time (in seconds) of each of the algorithms, and in the fourth column, one can see the average difference between the obtained EPL and the optimal EPL (referred to as the optimality gap as obtained by the ES procedure).

As seen in the website size, while the average running time of the ES and the AB&B methods grow very rapidly, the growth rates of the greedy method and the CE are much more moderate. Note that the relative optimality gaps obtained using all three heuristic methods decreased in the problem dimension as there is more combinatorial space for the heuristic to search for a better solution. This observation is of particular interest because it implies that the two faster heuristics can be applied to much larger websites at a reasonable computational cost.

Another observation is that the AB&B heuristic is dominated by the CE heuristic, both in terms of running time and optimality gap. The GD, CE, and ES constitute an efficiency frontier with respect to the average optimality gap and the average run time. It seems that in practice, the CE should be used for websites of moderate size, while the greedy heuristic, which obtained slightly higher EPL values, has a significant advantage in terms of running time, making it a good candidate method for larger websites. These observations are shown in Figure 6 where each method is depicted by the running time (using a logarithmic scale) and the average optimality gap.

Finally, we wish to demonstrate the effectiveness of the leading heuristics method by putting them in the context of the possible improvement that can be gained by deleting links from a websites with 24 pages (which is a typical size of a medium-size organization website). For each instance of the problem, we observed the ratio between the remaining absolute optimality gap and the total improvement that can be potentially gained by a complete optimization method. We refer to this ratio as a *distance gap*, defined as follow:

$$Distance\ Gap = \frac{Heuristic\ Solution - Optimal\ Solution}{Initial\ Solution - \ Optimal\ Solution}.$$

Figure 7 presents a histogram of the ratio of problem instances with respect to their distance gap (in predefined intervals). The solid lines represents the cumulative ratio of problem instances with a distance gap up to the corresponding category.

As seen, in more than 95% (respectively, 90%) of the instances, the CE heuristic (respectively, the greedy heuristic) obtained a solution to within 10% of the optimum. This implies that the two heuristics are capable of providing near optimal solution most of the time. Because these methods are much faster and scalable than any known complete solution method, it is possible to implement them online, for example, in personalized website browsing, when there is a need to make quick decisions.

Table I. Overall results – average of runs

Dimension (no of pages)	Heuristic	Avg. time (s)	Avg. optimality gap
8 pages	Initial solution	–	50.93%
	ES (OPT)	29.93	0.00%
	Greedy	0.10	2.14%
	AB&B	9.83	1.39%
	CE	16.07	0.21%
16 pages	Initial solution	–	25.68%
	ES (OPT)	1522.01	0.00%
	Greedy	0.55	1.19%
	AB&B	186.41	0.88%
	CE	38.39	0.49%
24 pages	Initial solution	–	20.95%
	ES (OPT)	10512.09	0.00%
	Greedy	0.95	0.67%
	AB&B	1032.18	0.43%
	CE	55.93	0.27%

ES, exhaustive search; AB&B, approximated branch and bound; CE, cross-entropy.

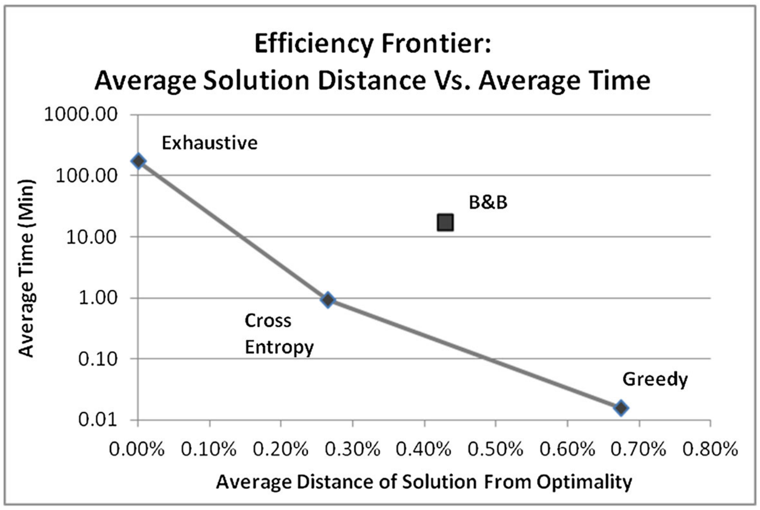

Figure 6. The Efficiency Frontier for runs of dimension 24

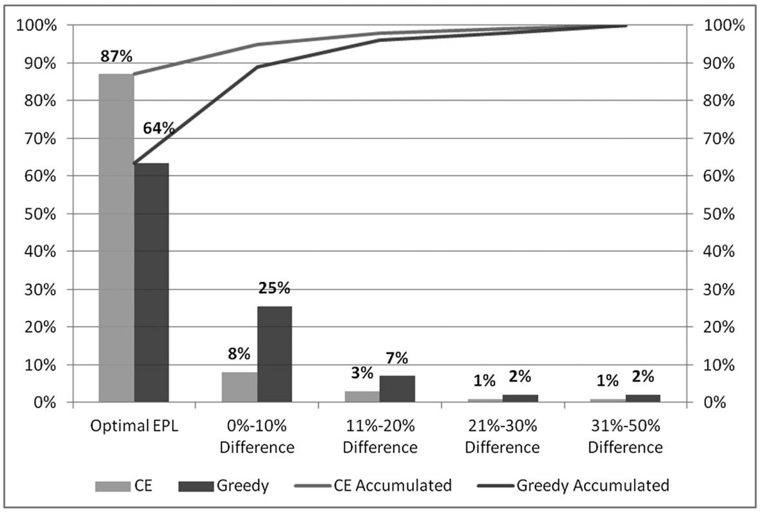

Figure 7. The Distance Gap for runs of dimension 24, both in CE and greedy. EPL, expected path length; CE, cross-entropy

6. Conclusions and future research

Effective linking within websites is crucial for obtaining a good user experience, including the reduction of the average search time until the users reach their desired content. Significant amount of search time can be saved by a better design of the website links' structure. Despite the fact that available information can be retrieved in each website on the users' preferences and their browsing behavior, both online and offline procedures for links' restructuring still remain a practical challenge and a subject for further research. It is clear that links' restructuring is only one aspect in webpage optimization that includes additional issues such as the webpage rank scoring, adaptability to search engines, ease-of-use, and other user experience issues. This research aims to cover only one of these aspects by proposing methods for 'smart' deletion of links, leading to a shorter website EPL. This task can be achieved relatively easily by deleting few links, without neither having to change the structure of the website, nor having to deal with issues of content understanding that might be even more challenging.

A series of experiments showed how deletion of website links can save the users up to 20% of clicks by using the proposed greedy and CE heuristics. Experiments that we executed on larger dimension websites (with 50, 100, or 200 pages) validate the overall conclusion that by disconnecting approximately 10% of the website links, one can often reduce more than 12% of the 'clicks' on average. This is an interesting observation because the conventional belief is that a denser link structure improves the visibility of the website and can reduce the browsing time. Implementing the link-deletion heuristics on relatively smaller website models (up to 50 pages that are common to medium and small organizations) showed that the greedy heuristic could be implemented as a real-time background application on the site, changing the linking layout of the site dynamically (e.g., by implementing a personalized website structure).

Future research direction includes among other issues not only the deletion of links but also an addition of links when required, as well as other user-experience factors. While this research considered each website as a single entity, future research should examine a larger structure of linked websites, allowing the heuristic to run quickly on a portion of the site in parallel, while enabling the user to move efficiently between the various networked sections. Another potential research direction can be focused on the use of fixed-order Markov chains to represent the user's behavior within the website. While one can look at a Markov chain of order 1 and derive a discrete-time stochastic matrix to represent the website movement of a user, another approach can consider a continuous-time Markov process. The latter modeling approach does not rely only on the current location of the user, but also on previous locations. Because such a modeling scheme is more complicated, one can alternatively consider a site containing *N* pages that can be represented by up to *N* discrete-time Markov processes, each of which represents a cluster or class of users with similar surfing behavior.

Finally, let us note that the use of browsing data as one out of many diverse customer-experience data sources is part of an ongoing effort in quality engineering to address modern application of QoS in an increasingly complex systems[33,34].

References

1. Jiang W, Au T, Tsui KL. A statistical process control approach to business activity monitoring. *IIE Transactions* 2007; **39**(3):235–249.
2. Liang CC, Wu PC. Internet-banking customer analysis based on perceptions of service quality. *Total Quality Management & Business Excellence* 2014:1–19.
3. Barnes M. Ralph, Motion and Time Study: Design and Measurement of Work (7th Edition). John Wiley, 1980.
4. Lawrence S. Aft, Work Measurement and Methods Improvement. Interscience, 2000.
5. Goldstain O, Ben-Gal I, Bukchin Y. Remote Learning for the Manipulation and Control of Robotic Cells. *The European Journal of Engineering Education* 2007; **32**(4):481–494.
6. Nielsen Website, http://www.nielsen.com/us/en/insights/news/2010/what-americans-do-online-social-media-and-games-dominate-activity.html, [08-02-2010].
7. Yarow J., Here's what people do on the internet all day, *Business Insider Website*, http://www.businessinsider.com/heres-what-people-do-on-the-internet-all-day-2014-5, [05-21-2014].
8. Murat AB, Ismail HT, Murat D, Ahmet C. Discovering better navigation sequences for the session construction problem. *Data & Knowledge Engineering* 2012; **73**:58–72.
9. Hussein W, Gharib TF, Ismail RM, Mostafa MGM. An efficient framework based on usage and semantic data for next page prediction. *Intelligent Data Analysis* 2015; **19**(6):1377–1389.
10. Xie Y, Yu SZ. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Transactions nn Networking* 2009; **17**(1):54–65.
11. Montgomery AL, Srinivasan SLK, Liechty JC. Modeling online browsing and path analysis using clickstream data. *Marketing Science* 2004; **23**(4):579–595.
12. Zukerman I, Albrecht DW. *Predictive statistical models for user modeling, User Modeling and User Adapted Interaction* 2001; **11**(1-2):1–5.
13. Rubinstein R. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computer in Applied Probability* 1999; **2**(2):127–190.
14. Perkowitz M., Etzioni O., Adaptive web sites: an ai challenge, in: Proc. IJCAI-97, Nagoya, Japan, 1997.
15. Perkowitz M, Etzioni O. Towards adaptive web sites: conceptual framework and case study. *Artificial Intelligence* 2000; **118**(1-2):245–275.
16. Mobasher B, Jain N, Han EH, Srivastava J. Web mining: Pattern discovery from world wide web transactions. In *Technical Report* TR-96050. Department of Computer Science, University of Minnesota: Minneapolis, 1996.
17. Forsati R, Moayedikia A, Shamsfard M. An effective web page recommended using binary data clustering. *Information Retrieval Journal* 2015; **18**:167–214.
18. Ben-Gal I. On the Use of Data Compression Measures to Assess Robust Designs. *IEEE Transactions on Reliability* 2005; **54**(3):381–388.
19. Jiang T, Tuzhilin A. Improving personalization solutions through optimal segmentation of customer bases. *IEEE Transactions on Knowledge and Data Engineering* 2009; **21**(3):305–320.
20. Baraglia R, Silvestri F. Dynamic personalization of web sites without user intervention. *Communications of the ACM* 2007; **50**(2):63–67.
21. Saad S. Z., Web Personalization based on Usage Mining, The 3rd BCS IRSG Symposium on Future Directions in Information Access, 2009, pp. 15-21.
22. Jalali M, Mustapha N, Sulaiman M, Mamat A. WebPUM: a web-based recommendation system to predict user future movements. *Expert Systems with Applications* 2010; **37**(9):6201–6212.
23. Saad SZ, Kruschwitz U. Applying Web Usage Mining for Adaptive Intranet Navigation, Multidisciplinary Information Retrieval, LNCS, Vol. **6653**. Springer Berlin Heidelberg: Berlin, 2011; 118–133.
24. Goyal P, Goyal N, Gupta A, Rahul TS. Designing self-adaptive websites using online hotlink assignment algorithm. *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, ACM* 2009:579–583.
25. Chen M, Ryu YU. Facilitating effective user navigation through web site structure improvement. *IEEE Transactions on Knowledge and Data Engineering* 2013; **25**(3):571–588.
26. Jadhav SW, Bhale NL. Optimized website structure improvement for effective user navigation. *International Research Journal of Engineering and Technology (IRJET)* 2015; **2**(9).
27. Bestavros A, Cunha C. Server-initiated document dissemination for the WWW. *IEEE Data Eng. Bull* 1996; **19**(3):3–11.
28. Zukerman I, Albrecht DW, Nicholson AE. UM99 - Proceedings of the Seventh International Conference on User Modeling. In Predicting Users' Requests on the WWW. Banff: Canada, 1999; 275–284.
29. Yin PY, Guo YM. Optimization of multi-criteria website structure based on enhanced tabu search and web usage mining. *Applied Mathematics and Computation* 2013; **219**:11082–11095.
30. Paul N, Kumar A. Adding extra links and repairing existing links in websites. *International Journal of Mechanical Engineering and Information Technology* 2014; **2**(9):746–753.
31. Singh H, Kaur P. A survey of transformation based website structure optimization models. *Journal of Information and Optimization Sciences* 2015; **35**:5–6 529-560. doi:10.1080/02522667.2014.961802.
32. Barabasi AL, Albert R, Jeong H. Scale-free characteristics of random networks: the topology of the World Wide Web. *Physica A* 2000; **281**(1):69–77.
33. Brombacher A, Hopma E, Ittoo A, Lu Y, Luyk I, Maruster L, Ribeiro J, Weijters T, Wortmann H. Improving product quality and reliability with customer experience data. *Qual. Reliab. Engng. Int.* 2012; **28**:873–886. doi:10.1002/qre.1277.
34. Kenett RS, Harel A, Ruggeri F. Controlling the usability of web services, international journal of software engineering and knowledge engineering. *World Scientific Publishing Company* 2009; **19**(5):627–651.

Authors' biographies

Zohar Postelnicu accomplished his BSc (2008) and MSc (2013) degrees in the Department of Industrial Engineering at Tel Aviv University, the latter under the supervision of Professor Irad Ben-Gal and Professor Tal Raviv. Zohar is currently working as a Product Manager at Google, located in Mountain View, CA, USA.

Tal Raviv is a member of the Industrial Engineering Department, Iby and Aladar Fleischman Faculty of Engineering, Tel Aviv University, Israel. He holds a BA from the Eitan Berglas School of Economics, Tel Aviv University (1993); an MBA from the Recanati School of Business, Tel Aviv University (1997); and a PhD in Operations Research from the William Davidson Faculty of Industrial Engineering and Management, Technion–Israel Institute of Technology, Haifa (2003). He spent 2 years (2004–2006) as a postdoctoral fellow in the Sauder School of Business at University of British Columbia, Vancouver, Canada. His current main research interest is in transportation and logistics with a focus on sustainable logistics and shared mobility systems.

Irad Ben-Gal is a professor and former chair in the Department of Industrial Engineering at Tel Aviv University, currently visiting the Management Science and Engineering Department at Stanford University. His research interests include applied probability, machine learning and information theory applications to industrial and service systems. He wrote and edited five books, published more than 80 scientific papers and received several best papers awards. Irad is the former chair of the Quality Statistics and Reliability society at INFORMS and a member in the Institute of Industrial Engineers, the European Network for Business and Industrial Statistics and the International Statistical Institute. He is a Department Editor in Institute of Industrial Engineers Transactions and serves in the Editorial Boards of several other professional journals. Prof. Ben-Gal supervised dozens of graduate students, led many R&D projects and worked with companies such as Intel, Applied Materials, GM, Nokia, Oracle and SAP. Irad is a co-founder and an active chairman of Context-Based 4casting (C-B4), a software startup that provides automated predictive analytics solutions.