Chapter 6

Problem detection phase

In Chapters 4 and 5 we observed people in “non-web” situations and on the WWW, respectively. We observed that the participants in our study faced some problems in reaching their goals. In Section 6.1 we give an overview of website problems web visitors face to reach their goals. We give an overview of search behavior on the WWW in Section 6.2 and in “non-web” situations in Section 6.3. Then, in Section 6.4, we address the first research issue ‘What can we learn from searching in a “non-web” setting vs. searching in a web setting for improving searching on the WWW?’. Finally, we develop in Section 6.5 an approach to reduce the website problems and to accommodate the website visitors’ search behavior. We discuss this in the next chapters.

6.1 Website problems

The results of Chapter 5 showed that website visitors regularly encounter problems in finding the information they look for. These problems can cause irritations to participants, who consequently will leave the website unsuccessfully. Below we give a list of website problems that the participants in our study encountered (see Section 5.6).

1. Web visitors do not get the expected information behind links or menu items.

2. Web visitors face 404 error pages.

3. The navigation support is not on the expected location of the website.
4. Web visitors can not find a search option on the visited website.

5. Web visitors get lost in the structure of the website because they do not know where they are in the website.

6. Web visitors cannot guess the URL of the visited websites, because the URL is not chosen logically.

7. There are too many images or animations on the visited website which makes the website slow.

6.2 Search behavior on the WWW

Below we provide a list of search behaviors we observed on the WWW (see Section 5.6).

1. Web visitors may change their goal(s).

2. Web visitors predict the content behind a link or menu item.
   This behavior is related to website problem 1.

3. Web visitors use navigation aids on the visited websites.
   This behavior is related to website problem 3.

4. Web visitors orient on the website first before they proceed.
   This behavior is related to website problem 5.

5. Web visitors try to guess the URL of websites.
   This behavior is related to website problem 6.

6. Web visitors use search engines to realize their goals.
   This behavior is related to website problem 6.

7. Web visitors use the browser search option when the visited websites do not offer a search option.
   This behavior is related to website problem 4.

In this list we observe that website problems 2 and 7 are not related to any of the observed search behaviors on the WWW.
6.3 Search behavior in “non-web” situations

Below we provide a list of search behaviors we observed in “non-web” situations (see Section 4.5).

1. People’s goals may be vague and sometimes unpredictable. This behavior is similar to search behavior 1 on the WWW.

2. People are attracted by state triggers. This behavior is similar to search behavior 1 on the WWW.

3. People expect things at certain locations. This behavior is similar to search behavior 2 on the WWW. Note that this behavior can be related to search behavior 5 as well when one interprets a URL as a location.

6.4 Searching in “non-web” setting vs. searching in a web setting

Referring to our first research issue ‘What can we learn from searching in a “non-web” setting vs. searching in a web setting for improving searching on the WWW?’ in Section 1.2, we conclude that we observe similarities when we compare the results of searching in a “non-web” setting with searching on the WWW. The search results teach us that the search behaviors in “non-web” situations are the same as the search behaviors on the WWW. This implies that some rules, which are valid in “non-web” situations, also apply on the WWW. Natural search behavior should be possible on the WWW as well. To improve searching on the WWW, the WWW should support all search behaviors that are accepted in our daily life outside the WWW. The WWW should not show any obstacles to support behavior that is accepted elsewhere too. Below we give some examples of natural search behavior on the WWW.

1. Unpredictable goals
The airport is mainly used for traveling purposes. For example, people go to the airport when they are going on holiday, picking up some family or friends from the airport or taking someone to the airport. Some airports have nice restaurants with nice panoramic views of airplanes.
People may go to the airport to have diner with their friends and to enjoy the panoramic view. This is an unpredictable goal.

On the WWW this is not different. Web visitors may visit the Walt Disney World Resort\(^1\) website not solely because they want to plan their vacation to Disney World, but also to play a game or to watch a movie of their favorite Disney hero. In Figure 6.1 we can see that web visitors can play games and watch movies besides booking a holiday at the Walt Disney World Resort.

Figure 6.1: The website of Walt Disney World Resort. Web visitors can play games and watch movies besides booking a holiday.

2. **Attracted by triggers**

   Suppose someone invited a friend home for dinner. He wants to serve him lasagna, but he does not have all the ingredients at home to make

\(^{1}\)http://www.disneyworld.com.
the lasagna. He decides to go to the supermarket to buy the ingredients. On his way to the supermarket he sees a restaurant with nice pictures of Chinese food in the window. The smell of the nice food triggers his attention even more, and he finally decides not to make lasagna at home but to have dinner with his friend at this restaurant. This is an example of attraction by state triggers in a “non-web” situation.

On the WWW web visitors are attracted by similar things as in “non-web” situations, for example, by advertisements, colors, blinking text, images, animations, headers, font sizes, font types, movies, etcetera. In Figure 6.2 we see a screenshot of the games.com website. Several advertisements are incorporated in this website. Web visitors may be triggered by, for example, the CenterParcs advertisement on the right side of the website. A web visitor may book a vacation instead of playing a game.

3. Expectation on location
At the street market people expect certain things at certain location. They expect, e.g., at the Albert Cuyp street market (see Section 5.1), that the flower booth is the first booth on the right side of the market (see Figure 5.1). They expect the salesman, who is selling oranges, is on the left side of the street market.

On the WWW web visitors usually expect that the logo of the company is at the left top corner. They expect that clicking on the logo will take them to the homepage of the website (see Figure 6.3).

Taking into account that the above rules are valid in both situations, “non-web” and web situations, we develop an approach to optimize websites.

6.5 Approach

We now know the problems web visitors mostly face when they visit websites. We also know the natural search behavior of people on the WWW and elsewhere. We will use this knowledge to optimize websites, and to better support wild searching. To this purpose, we will use the following approach.

Some of the website problems that we detected can be solved by applying user interaction design patterns (in short: design patterns). A design
pattern\textsuperscript{2} is a formal way of documenting a solution to a common design problem [244]. Interaction design patterns are a way to capture optimal solutions to common usability or accessibility problems in a specific context [248]. They document interaction models that make it easier for users to understand an interface and accomplish their tasks [222, 230].

In Chapter 7 we formulate design patterns to overcome the observed website problems and for the observed web visitors’ search behavior. The design patterns will be the basic components to optimize websites, including support for wild searching. We collect existing examples of the formulated

\textsuperscript{2}The architect Christopher Alexander is the father of the Pattern Language movement in computer science. He and his colleagues originally developed \textit{patterns} in a groundbreaking book called “A Pattern Language: Towns, Buildings, Construction” [20] in 1977. He believed that he could create an entire pattern language that was greater than the sum of the individual parts by connecting related patterns, and by showing how they intertwine and affect one another.
Approach

Figure 6.3: The website of Fiat. Web visitors expect the logo on the top left corner.

design patterns for static website optimization.

In Chapter 8 we model the web visitor’s search behavior in terms of Markov decision problems (MDP). With the mathematical model we can analyze the web visitor’s click behavior, and consequently where he is looking at. We show how we can influence the web visitor’s goal, and hence his click behavior, with state triggers. We show that this model can be applied to the different categories which we used to observe people in Chapters 4 and 5. This model can be used to optimize websites for wild searching. We apply the model to some examples on the WWW and to “non-web” settings to show how the model works. We finally show how the model can be used to describe the web visitor’s wild search behavior.

In Chapter 9 we provide an overview of some techniques that are applied in practice for website optimization. We show some concrete examples of tools that use those techniques to optimize websites. The applicability of the techniques are discussed in this chapter.

In Chapter 10 we describe a model for dynamic website optimization through
autonomous management of design patterns. The observation results of Chapter 5 are used to estimate model parameters. Then the results of the estimate are analyzed to optimize websites.

In Chapter 11 we present the conclusions of our research and we make recommendations for future work.
Chapter 7

Solution: design patterns for static website optimization

In this chapter we provide solutions for the problems listed in Chapter 6. Section 7.2 describes the structure of a design pattern. In Section 7.3 we first formulate a design pattern for wild searching. Then in Section 7.4 we formulate design patterns for static website optimization to overcome the observed website problems and to support wild searching. These design patterns will support the observed web visitor’s search behavior on the WWW (see Section 6.2). The design patterns will be the basic components to optimize websites and to better support wild searching. Besides our empirical studies, we will collect existing examples of design patterns that are applied in practice for static website optimization. We reformulated some of the design patterns because in our opinion some of them could be formulated more precisely. In Section 7.5 we show by means of transition diagrams how design patterns may influence the success of wild searching.

7.1 Introduction

By optimizing websites one can provide better support for wild searching. The design patterns in this chapter are based on the results of our empirical studies in Chapter 4 and 5. In addition to these results, some projects were done by students [160, 217]. These projects were supervised by the author. In the project “Path breadcrumbs in practice” (see also Appendix D.2) we worked on the actual path followed by a visitor on a website. We wanted
to know whether it was possible to make the actual path followed by a web visitor visible and if we could provide the actual path in a usable way to provide backtrack facilities. This study showed that using path breadcrumbs is a possible way to follow the visitor’s path on a website. Path breadcrumb is also a benefit for the visitor as he can see which path he has taken to reach the current page. The disadvantage of path breadcrumbs is that there is limited space available on the screen to display all the steps taken by the visitor. Therefore, only a limited number of pages can be displayed in the path breadcrumb on the current page. This can be solved by a “more results” link by which the visitor can see his whole browsing history on the website. In this way we can provide with path breadcrumbs a means to backtrack.

In the project “User’s search and navigation behavior in non-profit domains” (see also Appendix D.1) we investigated the users’ search and navigation behavior in non-profit websites in the Netherlands and the U.S.A. A so called Total Review™ method was used by the consultancy company Cemit to advise organizations in making their website more effective. This method was most suitable for e-commerce websites and needed to be made applicable for non-profit organizations. The study tried also to assess how successful the websites are in facilitating users in reaching their goals, in order to reach the non-profits organizational goals. In the study we focused on two non-profit sectors: local government and charities. The study showed that the websites involved in this study should be optimized and made also accessible for users doing a wild search. This is important because the success of non-profit websites is mainly dependent on finding the desired information easily. An elaborate description of these projects is presented in Appendices D.1 and D.2.

Design patterns can be useful for both website visitors and web designers. When applying the design patterns, website visitors will be supported better in performing a wild search. This will result in website visitors spending more time on the optimized websites, and be more satisfied as they can realize more goals, especially the goals that are not well defined and the new goals.

The design patterns can be useful to web designers as well. Website owners usually have a specific target in mind or a specific goal for their web visitors. Web designers can use the design patterns to steer the web visitor in the right direction. In particular, for e-commerce websites (see Chapter 1) steering the web visitor in the right direction might have the objective, e.g.,
to sell a product or to make money by making the web visitor click on banners. It can also be important to make the web visitor enthusiastic for new things (creating new user goals or making web visitors deviate from their initial goals).

In the next sections we formulate design patterns for static website optimization that benefit both the web visitor and the web designer while taking into account the observed website problems. These design patterns will support the observed web visitors intended search behavior (see also 6.2) and improve wild searching.

7.2 Structure of design patterns

Many experts [231, 4, 248, 225, 2] structure a User Interface (UI) design pattern in 4 to 6 steps, which always include the following:

1. Problem (what),
2. Solution (how),
3. Context (where/when), and
4. Implementation (how).

The structure of Welie [231] consists of six different steps. If one wants to solve a problem, questions like when, how, and why to use the solution arise. We will follow Welie’s structure, because his word choice is in our opinion more natural and clear. Below we give an overview of the different steps and provide a short description of each step.

1. **Problem**
   In this step we give a concise description of the situation or website problem the web visitor is facing that potentially calls for the design pattern.

2. **Solution**
   In this step we give a concise description or solution of what this design pattern entails in addressing the website problem.

3. **Use when**
   In this step we describe conditions and considerations that help you know when to use the design pattern.
4. **How**
   In this step we describe how to implement this design pattern.

5. **Why**
   In this step we explain why the design pattern works; why it is an effective solution.

6. **More examples**
   In this step we provide more examples of how others have used the design pattern to create inspiration and a better sense of the design pattern.

7.3 **Design pattern for wild searching**

We discovered the following design pattern for wild searching.

1. **Problem**
   Natural search behavior, like wild searching, should be possible on the WWW as well. To improve searching on the WWW, the WWW should support all search behaviors, in particular wild searching, that are accepted in our daily life outside the WWW. Web visitors who lack knowledge or contextual awareness to formulate queries to realize their goal should also be able to search on the WWW.

2. **Solution**
   Create a website that support wild searching and design it through the perspective of your web visitors. Web visitors should be able to search in a natural way. Make the same information on the website as much as possible accessible in many ways (see example Figure 7.1).

3. **Use when**
   Wild searching should be made possible whenever it is possible, especially when you want to give your web visitors the feeling that they can search on your website in a natural way. If you have an e-commerce site then wild searching is a must as you want your web visitors to turn into customers. This is only possible when web visitors will find the product that they are looking for on your website.
4. How

(a) Make the same information on your website as much as possible accessible through various ways besides navigation (Note: this solution is related to the observed search behavior 1 on the WWW (web visitors may change their goal(s)). See Section 5.6.2):

i. Offer a search option on your website (see Section 7.4.5 for a design pattern). A search option is a must for medium to large sites. Web visitors who can not find the information through the menu or navigation (see, e.g., [231] for menu and navigation design patterns), may find it through a search option.

ii. Offer a tag cloud (see, e.g., [248] for a design pattern). Most used and popular tags are shown by different font size and weight to reflect the relative number of times each tag was used. With a tag cloud web visitors quickly get an overview of a large amount of information that has been tagged by
other web visitors and their popularity.

iii. Offer a search cloud. A search cloud is a large list of words. Like in a tag cloud each word in a search cloud is a link, and they are all different sizes. Each word is something that someone searched for when they came to your website. If a lot of web visitors searched for a certain word, then that word will appear bigger in the cloud. A search cloud gives web visitors a quick overview of what other web visitor searched for on the website. The design pattern for a search cloud is the same as for a tag cloud (see, e.g., [248] for a design pattern).

iv. Offer a news section (see, e.g., [231] for a design pattern). The news section could be implemented in a newsbox (see, e.g., [231] for a design pattern). In this section web visitors get a quick overview of the new information, additions, and updates on the website. Other news like press releases are also shown in this section.

v. Offer a site map (see, e.g., [231] for a design pattern). A site map functions as a table of contents of the website and as a navigation alternative. Web visitors can reach pages in one click. A site map gives an overview of all the available pages on the website.

vi. Offer a section related items (see Section 7.4.10 for a design pattern). The section related items is based on the topic of the page the web visitor is visiting. The section related items shows what is also available on the website about the current topic or pages related to the current topic. The section offers web visitors to see more about certain topics on the website that web visitors normally would not see if they would not search for it. The related items could be presented as hyperlinks (see also Section 7.4.1 for a design pattern for tooltips in hyperlinks), images, or videos (see also Section 7.4.9 for a design pattern to optimize a website for speed).

vii. Offer a recommendations section (see Section 7.4.11 for a design pattern). This section is a list of topics recommended by the site owner. The section offers web visitors the opportunity to change their mind and deviate from their initial goal.
It can make web visitors curious to see what the site owner has recommended.

viii. Offer a section with most viewed pages (see Section 7.4.12 for a design pattern). This section might be called, e.g., top 5 or top 10 or popular lately. The section gives web visitors an idea of which pages and topics where popular recently. It gives web visitors the opportunity to change their mind by following the mass.

ix. Offer a section called what’s new or latest addition (see Section 7.4.13 for a design pattern). The section shows web visitors the latest additions on the website. In that respect it shows an overlap with the news section (see, e.g., [231] for a design pattern). The section can make web visitors curious to see what is new on the website.

(b) Use tooltips (see Section 7.4.1 for a design pattern). It is a good habit to use tooltips for hyperlinks or menu items (see, e.g., [231] for menu design patterns). Tooltips give web visitors an idea what is behind a hyperlink or menu item. Tooltips help web visitors in their search. (Related to website problem 1, web visitors do not get the expected information behind links or menu items).

(c) Use snapshots (see Section 7.4.2 for a design pattern). Snapshots are useful when you want to give web visitors an impression or preview of the page or content behind a hyperlink or menu item (see, e.g., [231] for menu design patterns). Snapshots facilitate web visitors in their search. (Related to website problem 1, web visitors do not get the expected information behind links or menu items).

(d) Use local breadcrumbs (see Section 7.4.6 for a design pattern). Local breadcrumbs show the route from the homepage to the current page. Local breadcrumbs give web visitors insight into the hierarchical structure of the website which is helpful in the web visitors’ search. Web visitors understand where they are in relation to the rest of the website. (Related to website problem 5, web visitors get lost in the structure of the website).

(e) Use navigation support on the proper position in your website (see Section 7.4.4 for a design pattern). Web visitors from, for example, the western cultures expect the navigation support on
the left side. Take your target group into account when you are designing the navigation support. Web visitors will appreciate that because they have to spend little time to look for the navigation support. (Related to website problem 3, the navigation support is not on the expected location of the website).

5. **Why**
   A website which is optimized for wild searching makes it possible for web visitors to search in a natural way. For e-commerce sites an optimized website for wild searching enhances the chances to turn web visitors into customers. In general all websites owners want their visitors to come back and a high ranking in the major search engines. Wild searching can be helpful in this.

6. **More examples**

![Figure 7.2: An example of a ‘Latest addition’ section. From K2joom (http://www.k2joom.com/).](image)
Figure 7.3: An example of most viewed pages (here: top songs, top movies, and top tv shows.). From Apple (http://www.apple.com/).
Figure 7.4: An example of a ‘Recommendations’ section. From Drupal (http://www.drupal.org/).
7.4 Design patterns to overcome the observed website problems and to support wild searching

7.4.1 Use tooltips

We discovered this design pattern for website problem 1 (*web visitors do not get the expected information behind links or menu items*) which also supports wild searching (see 4b of Section 7.3).

1. Problem
   Web visitors do not get the expected information behind links or menu items.

2. Solution
   Use tooltip feature for links or menu items. Figure 7.5 shows a tooltip example.

![Figure 7.5: An example of a tooltip.](image)

3. Use when
   Use tooltips for all the links or menu items when it is not clear on which page the visitor will arrive when he clicks on the link or menu item.
4. **How**
A simple way to create tooltips is by using the “title” attribute of the HyperText Markup Language (HTML)\(^1\). Give a short description in the tooltip of what web visitors can expect when they click on the link or menu item (see, e.g., [231] for menu design patterns). Show the URL of the page to which the link points. Other techniques for creating tooltips are Javascript, AJAX, or CSS.

5. **Why**
Web visitors can get irritated when they find unrelated information behind a link or menu items. To avoid that web visitors are surprised by unexpected information behind a link or menu item one should use tooltips for links or menu items.

6. **More examples**

![Figure 7.6](http://www.1976design.com/blog/archive/2003/11/21/nice-titles/)

---

\(^1\)See the World Wide Web Consortium (W3C) at [http://www.w3c.org](http://www.w3c.org).
Design patterns to overcome the observed website problems

Figure 7.7: From http://www.google.com.

Figure 7.8: From http://bueltge.de/wp-bubble-tooltip-plugin/142/.
7.4.2 Use snapshots

We discovered this design pattern for website problem 1 (web visitors do not get the expected information behind links or menu items) which also supports wild searching (see 4c of Section 7.3).

1. Problem
   Web visitors do not get the expected information behind links or menu items.

2. Solution
   Use snapshots to give web visitors a preview in advance of the website behind a link or menu item.

3. Use when
   Use snapshots for all the links or menu items when it is not clear on which page the visitor will arrive when he clicks on the link or menu item. Snapshots are also nice when you want to give web visitors a preview or an impression in advance of the page or content behind a link or menu item.

4. How
   A simple way to create snapshots is by using the free tools of service providers (e.g., Snap\textsuperscript{2}). There are also add-ons available for browsers

\textsuperscript{2}http://www.snap.com.
Design patterns to overcome the observed website problems

like Mozilla FireFox\textsuperscript{3} that can be installed by web visitors. Web designers can provide a link to these add-ons. Other techniques for creating tooltips are Javascript, AJAX, or CSS.

5. **Why**

Web visitors can get irritated when they find unrelated information behind a link or menu items. To avoid this surprise, one should use tooltips for links or menu items.

6. **More examples**


Figure 7.10: An example of an audioshot.

Figure 7.11: An example of a movieshot.

Figure 7.12: An example of a mapshot.
7.4.3 Check your links

We discovered the following design pattern for website problem 2 (*web visitors face 404 error pages*).

1. **Problem**
   Web visitors face 404 error pages or links which are not working properly.

2. **Solution**
   Check your website regularly for dead links.

![Figure 7.13: An example of a Mozilla Firefox add-on to check dead links on webpages. This add-on uses colors to indicate which links are good and which are dead links.](image)

3. **Use when**
   Information on the Internet changes rapidly. It is therefore wise to check your website regularly for dead links, especially when you have external links.
4. **How**

There are several tools (e.g., link validators), software, and add-ons available to automate link checks on websites. A few examples are the linkchecker (see Figure 7.13) add-on for Mozilla Firefox\(^4\) and the online linkchecker dead-links.com\(^5\).

5. **Why**

Web visitors do not like error pages. They get irritated or annoyed when they face 404 pages or links that are not working properly.

6. **More examples**

![Dead-Links.com - Free Broken Link Checker](http://www.dead-links.com/)

Figure 7.14: An example of an on-line link validator spider (\texttt{http://www.dead-links.com/}).

---


\(^5\)http://www.dead-links.com/.
Design patterns to overcome the observed website problems 135

Figure 7.15: Example of the software AM-Deadlink to check links (http://aignes.com/deadlink.htm).

Figure 7.16: An example of an on-line link checker at W3C (http://validator.w3.org/checklink).
7.4.4 Pay attention to the position of the navigation support

The following design pattern (related to website problem 3, *the navigation support is not on the expected location of the website*) was discovered by experts [231, 4]. This design pattern also supports wild searching (see 4e of Section 7.3).

1. **Problem**
   
The navigation support is not on the expected location of the website.

2. **Solution**
   
   Use the left side or the top of the website for main navigation support for people from western cultures.

Figure 7.17: From Sony (http://www.sony.co.uk/hub/bravia/block/4).

3. **Use when**
   
   Web visitors from western cultures expect the navigation support on the left side or the top of the website. Therefore it is wise to put the navigation support on the left side or the top when you are designing a website for people from western cultures.

4. **How**
   
   Place the main navigation support on the left side or the top of the website. There are many ways to design the main navigation for your
Design patterns to overcome the observed website problems

website. The most common ones are the horizontal, vertical, or the inverted L menu. See, e.g., [248, 231] for main navigation design patterns.

5. Why
In western cultures people read from left to right, and from top to bottom. They therefore expect the navigation support on the left side or the top of the website.

6. More examples
See [231, 4] for more examples.
7.4.5 Offer search options

The following design pattern (related to website problem 4, web visitors can not find a search option on the visited website) was discovered by experts [231, 248]. In this design pattern we added some examples (see Section 7.4.5, 6 ‘More examples’) to show how search results can be presented better to support web visitors in their search. (See also Appendix D.1)

1. Problem
   Web visitors cannot find a search option on the visited website.

2. Solution
   Offer web visitors a search option.

![Figure 7.18: Offer a search option. From Apple (http://www.apple.com/itunes/).](http://www.apple.com/itunes/)

3. Use when
   Offer web visitors a search option, especially when you have a website with many pages and topics. A search option is a must for medium to large websites.
4. **How**

   Use a small rectangular area that contains the search functionality. Place this area in a prominent position on the web page. It may help web visitors when a search engine highlights the searched keywords and gives the relevance of the link, e.g., in percentages, or puts the most relevant link (see Section 7.4.1 for a design pattern) at the top. A few lines of description of what one might expect behind the link is essential.

5. **Why**

   A search option on websites makes it easy for web visitors to look up the information easily and quickly. A search engine provides web visitors an alternative way to lookup and access information on websites.

6. **More examples**

   ![Figure 7.19: An example of a search engine with highlighted keywords (here: website optimization and wild searching) and the relevance in the search results. From searchcloud.net (http://www.searchcloud.net/).](image)
Figure 7.20: This is an example where the search can be narrowed (see left side of the webpage). The web visitor also has the option to see more results per page (see upper right corner of the webpage). From Lonely Planet (http://www.lonelyplanet.com/).
Design patterns to overcome the observed website problems

Figure 7.21: This is an example where the search engine comes up with related concepts (see top of the webpage) besides the search results. This is another way to narrow the search. From Yahoo! Search (http://search.yahoo.com/).
7.4.6 Use local breadcrumbs

Many people talk about breadcrumbs (see, e.g., [231, 4, 248]), while they mean local breadcrumbs. We think that one should specify what breadcrumb to use as there are two types of breadcrumbs: path and local breadcrumbs. Our contribution in the breadcrumbs design pattern is that we separated the two types of breadcrumbs. We reformulated the breadcrumbs design pattern. The following reformulated design pattern is for website problem 5 (*web visitors get lost in the structure of the website*). This design pattern also supports wild searching (see 4d of Section 7.3) (See also Appendix D.1)

1. **Problem**
   Web visitors get lost in the website because they do not know where they are in the website due to a lack of the insight into the hierarchical structure of the website. Web visitors need to know where they are in relation to the rest of the website; they need insight into the hierarchical structure of the website.

2. **Solution**
   Give web visitors an opportunity to know where they are in the website and how to navigate from the home page to the current page.

3. **Use when**
   Use local breadcrumbs on medium to large websites, especially sites with a hierarchical information structure with more than 3 levels deep. Local breadcrumbs are useful to give the web visitor insight into the hierarchical structure of the website.

4. **How**
   The local breadcrumb is a static breadcrumb that displays the route from the home page (see, e.g., [231] for a homepage design pattern) to the current page. Each level in the local breadcrumb functions as a link to that level. The levels are often separated by a >, >>, | or \ sign. The current page should not be a link. The local breadcrumb should preferably be placed above the content area but below the page header. In order to distinguish between local breadcrumbs and path breadcrumbs (see Section 7.4.7 for a path breadcrumb design pattern) we suggest to place a hierarchy symbol behind the local breadcrumbs.
Design patterns to overcome the observed website problems

5. Why

A local breadcrumb shows the route from the home page to the current page. Local breadcrumbs should show the website hierarchy, not the web visitor’s history. They are useful because they show the current location. This helps web visitors understand where they are in relation to the rest of the website. Local breadcrumbs give the web visitor insight into the hierarchical structure of the website. Local breadcrumbs can be used as a secondary navigation aid. With local breadcrumbs web visitors have the possibility to access higher site levels with one
click.

6. **More examples**
   See [231, 4, 248] for more examples.
7.4.7 Use path breadcrumbs

The following reformulated design pattern is for website problem 5 (*web visitors get lost in the structure of the website*).
(See also Appendix D.1 and Appendix D.2)

1. **Problem**
   Web visitors get lost in the website because they do not know where they are in the website and how they arrived on the current page. Browser back buttons do not remember all the visited websites and consequently does not show always the route to the current page. Another drawback of the browser back button is that endlessly clicking back can be tedious. Web visitors need a means to step back and to see what route they have taken to arrive at the current page.

2. **Solution**
   Give web visitors an opportunity to know where they are in the website and how they arrived at the current webpage.

   ![Home ➤ Movies ➤ Movie Calendar](http://www.hollywood.com)

   Figure 7.24: From Hollywood.com (http://www.hollywood.com).

3. **Use when**
   Use path breadcrumbs on medium to large websites, especially sites with a hierarchical information structure with more than 3 levels deep. Path breadcrumbs are important when you want to visualize the route the web visitor has taken to arrive at the current page.

4. **How**
   Show the web visitor’s browsing history in the website, i.e., the actual route the web visitor has taken to reach the current page. So, the path breadcrumb is a dynamic breadcrumb instead of a static one (see Section 7.4.6 for a local breadcrumb design pattern). Each level in the path breadcrumb functions as a link to that level. The levels are often separated by a >, >>, | or \ sign. The current page should not be a link. Use a label “More results” if the path breadcrumb becomes
too long (see Figure 7.26). The path breadcrumb should preferably be placed above the content area but below the page header. In order to distinguish between local breadcrumbs and path breadcrumbs we suggest placing a foot steps symbol behind the path breadcrumbs like in Figure 7.25. In this way web visitors will recognize that the breadcrumbs are path breadcrumbs and that they give information about the route the web visitor has taken to arrive at the current page.

![Home ▶ Movies ▶ Movie Calendar](image)

Figure 7.25: A foot steps symbol behind the breadcrumbs indicates that we are dealing with path breadcrumbs.

5. **Why**

A path breadcrumb shows the actual route that the web visitor has taken to arrive at the current page. It visualizes the web visitor’s browsing history on the website. In this way, path breadcrumbs provide the web visitor a means to step back and to see what route he has taken to arrive at the current page.

6. **More examples**

![Master's website - PHP version (session)](image)

Figure 7.26: This is an example of a path breadcrumb implemented by one of our students [217]. The label “more results” is used here because the path breadcrumb became too long. This example shows that the web visitor visited some pages, and finally returned to the PHP main page again.
Design patterns to overcome the observed website problems

Figure 7.27: An example of a browser history in Mozilla Firefox v3.6.3 (http://www.mozilla.org) which could be used as a path breadcrumb. A counter displays how many times a page is visited. This is a drawback as you can not see in which order you visited this page. Nevertheless, this browser history gives you an impression of the path you have followed to arrive at the current webpage.

Figure 7.28: Another example of a browser history in Mozilla Firefox v3.6.3 (http://www.mozilla.org) which could be used as a path breadcrumb. Here the history is displayed in a sidebar (like most of the popular browsers do) which is more convenient to be used as a path breadcrumb. The web visitor can choose from the different view options (e.g., view by the last visited webpage) to display the webpage he visited.
Figure 7.29: An example of a history list on an iPhone (http://itunes.apple.com/ca/app/fonolo/id348228086?mt=8) which could be used as a path breadcrumb. From the history the user can see to whom he made the phone calls and when.
7.4.8 Use logical URLs

We discovered the following design pattern for website problem 6 (web visitors cannot guess the URL of the visited websites) (see also Appendix D.1).

1. Problem
   Web visitors cannot guess the URL of the visited websites, because the URL is not chosen logically.

2. Solution
   Choose a URL which is logical or which reflects the website’s title.

Figure 7.30: Choose a URL which is logical. From Youtube (http://www.youtube.com).

3. Use when
   A logical URL should be chosen whenever it is possible, especially when you want to be easily found on the WWW.

4. How
   A logical URL looks like www.[website title].[top-level domain]. The website title is the title of the website and the top-level domain is the last part of an Internet Domain Name.

5. Why
   When web visitors visit a website they first try to guess the URL of the website. A logical URL makes it easier for web visitors to guess the website URL. If they fail to guess the URL, they will try search engines to find the website. It is likely that web visitors will arrive at other websites, rather than the intended one when using search engines.

6. More examples
Figure 7.31: This is an example of the webshop Wehkamp from the Netherlands. It uses .nl as the top-level domain to indicate that the shop is located in the Netherlands. From Wehkamp (http://www.wehkamp.nl).

Figure 7.32: Paris Hilton is using her name in her website. From parishilton (http://www.parishilton.com).
Figure 7.33: This is an example of Massachusetts Institute of Technology (MIT). They are using the .edu top-level domain indicating that they are an educational institution. From MIT (http://www.mit.edu).
7.4.9 Optimize website for speed

We discovered the following design pattern for website problem 7 (there are too many images or animations on the visited website which makes the website slow).

1. Problem
   There are too many images or animations on the visited website which makes the website slow.

2. Solution
   The download speed of a webpage depends on many things, like images, coding of the webpage, bandwidth, internet traffic, etc. Keeping the use of images to a minimum contributes to optimizing speed. There are several tools available on the Internet to check the download speed of your webpage. Use those tools to check where you can gain speed by optimization. See Figure 7.34.

Figure 7.34: The website of iWebtool provides a service called “Website Speed Test” to test the download speed of a webpage. From iWebtool (http://www.iwebtool.com).
3. **Use when**
   Reduce images and animations on webpages when you want to improve the performance and download speed of your website.

4. **How**
   Do not use unnecessary images or animation on your website. Keep the size of the images and animations as small as possible when you use images or animations. Use compression methods to keep the image size small. Test the download speed of your website with one of the available tools on the Internet. Check on which point(s) you can gain speed by optimization, and optimize the webpage at that point(s).

5. **Why**
   Web visitors do not like slow websites \[162, 167, 138\]. When they have to wait too long for a webpage, they can get frustrated and irritated, and consequently leave the website.

6. **More examples**
Figure 7.35: The website of websiteoptimization provides a service called “Web Page Analyzer” to test the performance and download speed of a webpage. The output of the test gives a lot of information about the submitted webpage. From websiteoptimization (http://www.websiteoptimization.com).
Figure 7.36: On the website of “test everything!” one can do a lot of tests on webpages. This website uses services from other websites. From “test everything!” (http://tester.jonasjohn.de).
Figure 7.37: The website of pingdom provides, among others, a tool to test webpages. It tests, e.g., the download speed of the webpage, it gives the number of CSS, RSS including HTML files, and the number of images used on the webpage included their size. From pingdom (http://tools.pingdom.com).
7.4.10 Related items

We discovered the following design pattern that supports wild searching (see 4(a)vi of Section 7.3).

1. **Problem**
   Web visitors need to know what is more available on the website related to the visited topic.

2. **Solution**
   Offer a section of related items based on the topic the web visitor currently is visiting.

![Figure 7.38: An example of related items. Here it is called ‘Related products’. From Adobe (http://www.adobe.com).](image)

3. **Use when**
   Use the section of related items to give web visitors the opportunity to do wild search and to know what is more available on the website related to the topic they currently are visiting. In particular, this section is important for e-commerce and for medium to large sized websites.
4. How
Show in your navigation menu (see, e.g., [231] for menu design patterns) or at the end of the webpage a section of related items (e.g., related articles, products, galleries). The items in this section should be related to the current topic the web visitor is visiting. Give the section a name (e.g., related articles) from which it is obvious that the items are related to the current topic. Often this section is built dynamically from a database in which the content of the website is stored.

5. Why
The related items section offers web visitors to see more about certain topics on the website than what web visitors would normally see if they had not searched for the additional information explicitly. Web visitors may be triggered by the related items to visit the related topics. A related items section supports wild searching better.

6. More examples

Figure 7.39: An example of related items. Here it is called ‘Related Links’. From Azrul (http://www.azrul.com).
Design patterns to overcome the observed website problems

Figure 7.40: An example of related items. Here it is called ‘Related galleries’. From Wikipedia (http://en.wikipedia.org).

Figure 7.41: An example of related items. Here it is called ‘Related Articles’. From Travels.com (http://www.travels.com).
7.4.11 Recommendations

We discovered the following design pattern that supports wild searching (see 4(a)vii of Section 7.3).

1. Problem
Web visitors need to know what is recommended by the site owner or by other web visitors.

2. Solution
Offer a section with a list of recommended items (e.g., recommended links, articles or products).

![Figure 7.42: A recommendation example. From Youtube (http://www.youtube.com).](image)

3. Use when
Use a recommendation section to offer web visitors an opportunity to do wild search and when you want to show more topics from your website. A recommendation section is also convenient when you want to personalize a webpage by, e.g., a section called “Recommended for you”. This section is very suitable for e-commerce and medium to large sized websites.

4. How
Show a section with a list of recommended topics. Call the section, e.g., “We recommend”, “Recommendations” or “Recommended for you”.
5. Why
A recommendation section supports wild searching better. It offers web visitors the opportunity to change their mind and deviate from their initial goal. It can make web visitors curious to see what is behind the recommended link(s). With a recommended section web visitors get an opportunity to see more from a website than what they would normally see if they had not searched for the additional information explicitly.

6. More examples

Figure 7.43: A recommendation example. From iJoomla (http://www.ijooomla.com).
Figure 7.44: A recommendation example. From CNN (http://www.cnn.com).

Figure 7.45: A recommendation example. From Joomplace (http://www.joomplace.com).
7.4.12 Most viewed pages

We discovered the following design pattern which supports wild searching (see 4(a)viii of Section 7.3).

1. **Problem**
   Web visitors need to know which topics are popular lately.

2. **Solution**
   Offer web visitors a section with a list of popular items (e.g., articles, products or links).

3. **Use when**
   Use a section with popular items to offer web visitors an opportunity to do wild search. This section is also convenient when you want to show more topics from your website.

4. **How**
   Offer a section with a list of top 5 or top 10 most viewed webpages. Preferably this section should be listed at the top of the page so that

---

Figure 7.46: From Youtube (http://www.youtube.com).
web visitors will see this first. Give the section a name like, ‘Top 5’, ‘Top 10’, ‘Most viewed’, ‘Popular’ or ‘Popular lately’.

5. Why
A section with a list of popular items gives web visitors an idea of which pages and topics were popular lately. It gives web visitors the opportunity to change their mind by following the mass. This section contributes to a better wild search.

6. More examples

![Figure 7.47: From Linux Journal (http://www.linuxjournal.com).](http://www.linuxjournal.com)
Design patterns to overcome the observed website problems

Figure 7.48: From Sweets Lyrics (http://www.sweetslyrics.com).

Figure 7.49: From ServerWatch (http://www.serverwatch.com).
7.4.13 Latest addition

We discovered the following design pattern that supports wild searching (see 4(a)ix of Section 7.3).

1. **Problem**
   Web visitors need to know what the latest additions are to the website.

2. **Solution**
   Provide a section with the latest addition.

   - **Problem**
     Web visitors need to know what the latest additions are to the website.

   - **Solution**
     Provide a section with the latest addition.

   - **Use when**
     Offer a section with the latest addition when you want to inform your web visitors about the new items (e.g., topics, articles or products) you recently added to your website. This section is also useful when you want to offer web visitors an opportunity to do wild search. The section is highly recommended for blog websites and websites with many articles (e.g., websites for journals or magazines).

   - **How**
     Provide a section with a list of recently added items (e.g., topics, articles or products). Call it, e.g., ‘What’s new’, ‘Latest additions’, ‘New
Design patterns to overcome the observed website problems

... releases’. Preferably this section should be listed at the top of the webpage so that web visitors will see this first.

5. Why
A section with a list of recently added items gives web visitors insight into what was added recently to the website. It can make web visitors curious to see what is new on the website. It gives web visitors a reason to explore the website further. The section contributes to a better wild search.

6. More examples

Figure 7.51: From MSDN Magazine (http://www.msdn.microsoft.com).
Solution: design patterns for static website optimization

Figure 7.52: From iJoomla (http://www.ijoomla.com).

Figure 7.53: From The Karaoke Channel (http://www.thekaraokechannel.com).
7.5 Transition diagrams for the observed problems

In this section we use the problems listed in Chapter 6 and the provided solutions in this chapter to draw transition diagrams. We will model how appropriate use of design patterns changes the expectations of user success and the perspective on the problems that we identified. The transition diagrams visualize the relationship between the problems and the proposed solutions. In Chapter 8 we develop and show how mathematical models, in particular Markov processes, can support website optimization based on these transition diagrams. The transition diagrams in this chapter will help us to understand these mathematical models better.

The transitions diagrams are presented below. Each transition diagram starts with with two rectangles, the initial states. In the rectangles we give two opposite statements, e.g., ‘The website uses a link checker’ (related to a solution, the design pattern) and ‘The website does not use a link checker’ (related to the problem we observed). An arrow in the diagram presents the transition from one state to another. In the diagrams we see that the initial states transit to other states (ovals). These ovals are the effects of what is mentioned in the rectangles. Finally, this results in a state in which the web visitor did reach his goal or did not reach his goal.

Below we discuss each diagram briefly.

7.5.1 Use tooltips or snapshots

The transition diagram in Figure 7.54 is related to website problem 1 (web visitors do not get the expected information behind links or menu items). We see that web visitors can get the expected information behind the links or menu items when a website uses tooltips or snapshots for preview. Based on this state web visitors may decide to leave the website or to continue their search on the website if they have not realized their goal at this state. When they continue their search they may or may not realize their goal.

It might also happen that web visitors do not get the expected information behind the links or menu items although a website uses tooltips or snapshots for preview. From this state web visitors may decide to leave the website or to continue their search on the website. When they continue their search they may or may not realize their goal.

Three things may happen when a website does not use tooltips or snap-
shots for preview. First, web visitors get the expected information behind the links or menu items. Second, web visitors do not get the expected information behind the links or menu items. Third, web visitors have no idea what is behind the links or menu items. In the latter state web visitors may decide to continue their search on the website or they may decide to leave the website without realizing their goal. In all the three states web visitors may or may not realize their goal.

7.5.2 Check your links

The transition diagram in Figure 7.55 is related to website problem 2 (web visitors face 404 error pages). From this diagram we can see that web visitors do not get 404 error pages if a website uses a link checker. From
this state web visitors may decide to leave the website or to continue their search on the website. There is also a possibility that web visitors go back to the previous webpage. This transition is given with a dotted arrow in the transition diagram as this does not occur often. Web visitors may or may not realize their goal when they continue their search on the website.

Web visitors may face 404 error pages if a website does not use link checkers. From this state web visitors may decide to leave the website or to go back to the previous webpage. When the web visitors go back to the previous webpage they may decide to continue their search on the website or to still leave the website. In the latter state web visitors will not realize their goal. When they continue their search they may or may not realize their goal.
7.5.3 Pay attention to the position of the navigation support

![Transition diagram](image)

Figure 7.56: Transition diagram for website problem 3 and the corresponding solution.

The transition diagram in Figure 7.56 is related to website problem 3 (the navigation support is not on the expected location of the website). From this diagram we can see that web visitors may or may not mind the location of the navigation support, either a website has or has not the navigation support on the top or on the left side. If web visitors do mind the location of the navigation support then they may expect or may not expect this on the top or on the left side. In both cases web visitors may decide to continue their search on the website or to leave the website. In the latter state web
visitors will not realize their goal. When they continue their search they may or may not realize their goal.

### 7.5.4 Offer search options

![Transition diagram for website problem 4 and the corresponding solution.](Image)

Figure 7.57: Transition diagram for website problem 4 and the corresponding solution.

The transition diagram in Figure 7.57 is related to website problem 4 (web visitors cannot find a search option on the visited website). From this diagram we can see that either a website offers or does not offer a search option, web visitors may or may not look for a search option. Web visitors can or cannot find a search option if a website offers a search option. If web
visitors can find a search option they may decide to continue their search on the website or to use the search option. After using the search option web visitors may decide to continue their search on the website or to leave the website. When they continue their search they may or may not realize their goal.

It is obvious that web visitors will not find any search options if a website does not offer any search possibilities. Web visitors may decide to continue their search on the website or to leave the website. If they leave the website then they will not realize their goal. They may or may not realize their goal if they continue their search.

### 7.5.5 Use path or local breadcrumbs

The transition diagram in Figure 7.58 is related to website problem 5 (*web visitors get lost in the structure of the website*). From this diagram we can see that if a website offers breadcrumbs this may be local or path breadcrumbs. Web visitors know where they are in the website if a website uses local breadcrumbs. Web visitors know or do not know the path they took to the current webpage if a website uses local breadcrumbs.

Web visitors know or do not know where they are in the website if a website offers path breadcrumbs. Web visitor know for sure the path they took to the current webpage if a website offers path breadcrumbs.

There are four options if a website does not offer any breadcrumbs:

1. The web visitors know where they are in the website.
2. The web visitors do not know where they are in the website.
3. The web visitors know the path they took to the current webpage.
4. The web visitors do not know the path they took to the current webpage.

From all the four states web visitors may decide to continue their search on the website or to leave the website. They may or may not realize their goal if they continue their search. If they leave the website then they will not realize their goal.
Transition diagrams for the observed problems

Figure 7.58: Transition diagram for website problem 5 and the corresponding solution.

7.5.6 Use logical URLs

The transition diagram in Figure 7.59 is related to website problem 6 (*web visitors cannot guess the URL of the visited websites*). From this diagram we can see that web visitors can guess the URL if a website uses a logical URL. There is also a possibility that some web visitors cannot guess the URL. These web visitors will use either search engines or their bookmarks to find the website. This is also the case when a website does not use logical URLs. Web visitors will not realize their goal if they cannot guess the URL. In the other cases they may or may not realize their goal.
Figure 7.59: Transition diagram for website problem 6 and the corresponding solution.

7.5.7 Optimize website for speed

The transition diagram in Figure 7.60 is related to website problem 7 (there are too many images or animations on the visited website). From this diagram we can see that either a website is or is not optimized for speed, web visitors may or may not mind the speed of the website (i.e., the time that it takes to display a webpage on the web visitor’s computer screen). Either web visitors mind or do not mind, they may continue their search or leave the website. They may or may not realize their goal if they continue their search. If they leave the website then they will not realize their goal.
Figure 7.60: Transition diagram for website problem 7 and the corresponding solution.
7.6 Conclusion

In this chapter we provided solutions (see Section 7.3 and 7.4) for the website problems listed in Chapter 6. We formulated design patterns for static website optimization. The design patterns can support the observed web visitors search behavior (see also Section 6.2) on the WWW and can improve wild searching.

In Section 7.5 we drew transitions states to visualize the detected problems and the solutions proposed in our design pattern list, and their relationship. In Chapter 8, we develop and show how mathematical models can support website optimization. The transition states in Section 7.5 will help us to understand these mathematical models better.
Chapter 8

Mathematical formulation of search behavior

8.1 Introduction

Search behavior of people is a complex process. In order to quantify this so that a systematic study of search behavior is possible, one can model search behavior in the framework of Markov decision problems. Markov decision problems are a widely used class of models that have been used particularly in behavioral ecology, communication models, inventory management, and economic planning and consumption models (see [192] and [49]). In this chapter we will introduce the basic components of a Markov decision problem and discuss some mathematical and notational subtleties in order to formulate search behavior in terms of a Markov decision problem. At the end of the chapter, we will apply the formulated model on some examples to show how the model works. These examples are meant to familiarize oneself with Markov decision models. This will be generalized in the Chapter 10 to deal with autonomous management of design patterns for website optimization.

8.2 Markov Decision Problems

The results of the empirical studies in Chapter 4 and 5 refer to sequential decision problems that can be modeled by ‘Markov decision problems’
180 Mathematical formulation of search behavior

(MDPs)\(^1\). In a Markov decision problem we are given a dynamical system whose state may change over time. This is exactly what we saw in Chapter 4 and 5; the actions that the users take are based on the information of the state they find themselves in and their expectations of the outcome of the actions (e.g., users expect certain information behind certain links, see also Section 5.6.1). Every action taken by the user at a certain point in time brings the user to another state. The new state provides the user new information and experience. Users get new ideas, new goals, or they even may change their goal. Combined with the information from the previous state the user may take another action of which he expects that it will bring him potentially closer to his goal. These steps continue until the goal is reached or the user states another goal or the user stops. The next section formalizes this idea.

8.3 Mathematical framework

In this section, we provide a formal mathematical description of Markov decision problems that will serve as a basis for describing search behavior and website optimization. A Markov decision problem is characterized by the tuple \((S, A, P, r)\). The set \(S\) describes the state, i.e., the information that a user is provided with, of the system that is being modeled. Suppose that at time \(t\) the user is given the information \(s_t \in S\). Then we can decide to take action \(a_t \in A\) to influence the system to move closer to his goals. When an action \(a_t\) is chosen based on the observed state \(s_t\), two things happen. First, the user incurs some reward \(r_t\) (i.e., utility to the user) based on the state and the action, denoted by \(r_t(s_t, a_t)\). Second, due to action \(a_t\), the state of the system moves to a new state \(s_{t+1}\) according to some probability law \(P_t\), that depends on the state and action as well. In this new situation, the user is faced with a similar problem structure as before; he observes a state \(s_{t+1}\) and has to choose an action \(a_{t+1}\) that brings him closer to his goal, after which the system moves on according to the probability law \(P\). These steps continue until the goal is reached or the user stops (see also Figure 8.1).

The abstract framework above is sufficiently rich to model search behav-

---

\(^1\)Note that Markov decision theory is a relatively new field of which the roots can be traced back to the 1950s [234]. MDPs are an extension to the much older field of Markov chains, of which initial results were published in the 1900s [149].
ior and support website optimization. By giving the elements of the Markov decision problem different interpretations, different situations of search behavior and website optimization can be modeled, both in “non-web” situations and in web situations. For example, in order to model shopping behavior, the system of the Markov decision problem could be taken to be a supermarket. The set $S$ could represent the different products that are present in your shopping cart, and $A$ the actions that apply to the shopping cart (e.g., add products or remove products). In this case, based on your shopping cart and your decision, the probability law $P$ is not random and completely deterministic, and leads to a reward (i.e., a sum of utilities that could represent user satisfaction and monetary costs) that is modeled by $r$.

In Section 8.4.1 (the supermarket example) we formalize this example to illustrate the concept of Markov decision problems. In Section 8.4.2 (the Free Record Shop example) we use the Markov decision problem to study search behavior of users, in particular, we show how the framework can formalize
the situation in which users have parallel goals.

In Section 8.4.3 (the look and click behavior example) we show how the framework can be used to study search behavior in web situations. To this purpose, we take a website as the system in our framework. In this case, the set $S$ represents the different webpages of the website, and $A$ denotes the action of clicking on a particular hyperlink on a webpage. The probability law $P$ is again a deterministic function, that translates the action into a new state. The reward $r$ represents the utility of the information found on the webpage, i.e., did the user find the particular information he is looking for on the webpage or not. We illustrate, within this framework, how navigation aids can lead to goal changes and to better support wild search.

The three examples (the supermarket, the Free Record Shop, and the look and click behavior example) above illustrate how Markov decision problems can be used to model and study search behavior in different settings. In Chapter 4 we observed the search behavior of people in “non-web” situations. The supermarket (see Section 8.4.1) example fits well in that context. In Chapter 5 we observed the search behavior of people on the WWW. In both settings (web and “non-web”) we used categories (see Section 4.2.6) to score the observation results. We randomly chose for the parallel goals (trade-off) category to illustrate the concept of Markov decision problems. We elaborate on this in the Free Record Shop Example (see Section 8.4.2). In the look and click behavior example we zoom in on the WWW. We use the categories navigation aids and goal changes and show how Markov decision problems can also be used to influence search behavior. With this example we also show that the mathematical model is applicable for wild searching.

8.4 Applications of Markov decision problems

8.4.1 Markov decision problem for shopping behavior in a supermarket

Imagine a user that is faced with the problem of buying items from a shopping list in a supermarket. He starts out with an empty shopping cart and needs to decide which products he is going to buy. Every decision to buy a product brings with it a certain cost, expressed in the time to obtain the product, the effort the user spends obtaining the product, and the price of the product. Hence, the user can evaluate the cost of every decision as a
The objective of the user is to satisfy the demands described by the items on the shopping list. However, the items on the shopping list can be described in various ways. The list may contain specific items (e.g., a certain product of a specific brand), but can have less specified items as well (e.g., a product group). The latter group of items can therefore be substituted by multiple products in the supermarket. Consequently, the decision to buy a certain product must be weighted against the cost that the user attaches to the decision to buy it. Moreover, the decision cannot be seen in isolation, since a decision now may affect a decision in future (e.g., due to constraints on the budget). The user is thus faced with a sequential decision problem of buying optimally given his shopping list.

In order to study behavioral aspects of the user, we formulate the problem as a Markov decision problem. Assume that the supermarket sells \( M \in \mathbb{N} \) different products. The so-called state space or problem space \( S \) denotes all the combinations of products that the user can buy, and is defined by the power-set

\[
S = \mathcal{P}(\{1, \ldots, M\}). \tag{8.1}
\]

Note that a state \( s \in S \) can be seen as the content of the shopping cart. The process of buying items can be modelled in discrete time. Every time the user takes an action the time is incremented by one unit of time. Thus, the situation that the user has an empty shopping cart at time \( t = 0 \) is represented by state \( s_0 \in S \) given by

\[
s_0 = \{ \} = \emptyset. \tag{8.2}
\]

At every time \( t \) the user has to make a decision. He can choose to add a product to his cart, to remove a product from his cart, or to do nothing. The actions available to the user are given by the so-called action space \( A \) defined by

\[
A = \{-M, \ldots, 0, \ldots, M\}, \tag{8.3}
\]

where action \( a_t \in \{1, \ldots, M\} \) means that the user decides to buy product \( a_t \) at time \( t \) and adds this product to his shopping cart. Action \( a_t \in \{-M, \ldots, -1\} \) means that at time \( t \) the user decides not to buy product \( a_t \) and removes the product from his shopping cart. Action 0 means that the user decides to buy nothing. The action set may also depend on the state of the system. We might want to model that the user does not buy the same product multiple times, and that he cannot remove a product that is
not in his shopping cart. Suppose that the user has \( n \) products \( s^1, \ldots, s^n \) in his shopping cart. Then the action space \( \mathcal{A}_s \) when the system state is \( s = \{s^1, \ldots, s^n\} \in S \) is given by
\[
\mathcal{A}_s = \{-s^1, \ldots, -s^n\} \cup \{0\} \cup \{1, \ldots, M\} \setminus \{s^1, \ldots, s^n\}.
\] (8.4)

Here, the first set in the definition of \( \mathcal{A}_s \) denotes that one can only remove products \( s^1, \ldots, s^n \) from the shopping cart. The second set is the action when the user buys nothing. The last set denotes all the products minus the products already in the shopping cart, and models that no product can be bought multiple times.

At every time \( t \) the user makes a decision the state of the system changes. The change in the system can be described by a transition law \( \mathbb{P} \) which relates the current state \( s_t \), the chosen action \( a_t \), and the next state \( s_{t+1} \) to each other. Since, in our case the contents of the shopping cart is known with certainty the transition law is deterministic, i.e., \( \mathbb{P} \) is either 0 or 1. Thus,
\[
\mathbb{P}(s_t, a_t, s_{t+1}) = 1 \text{ if and only if } \begin{cases} s_{t+1} = s_t \setminus \{a_t\} & \text{if } a_t \in \{-M, \ldots, -1\}, \\ s_{t+1} = s_t & \text{if } a_t = 0, \\ s_{t+1} = s_t \cup \{a_t\} & \text{if } a_t \in \{1, \ldots, M\}. \end{cases}
\] (8.5)

where \( s_t \in S \) is the content of the shopping cart at time \( t \), \( a_t \in \mathcal{A}_{s_t} \) is the action the user has chosen at time \( t \), and \( s_{t+1} \in S \) is a new content of the shopping cart after the decision. To illustrate the transition law, suppose that at time \( t = 0 \) the state of the system is \( s_0 = \{\} \). Suppose that the user takes the following decisions, \( a_0 = 1, a_1 = 5, a_2 = 0, a_3 = -1, \) and \( a_4 = 4 \), then it follows from the transition law that the subsequent states are \( s_1 = \{1\}, s_2 = \{1, 5\}, s_3 = \{1, 5\}, s_4 = \{5\}, \) and \( s_5 = \{4, 5\} \). These states describe that the user buys product 1, product 5, does nothing, removes product 1, and finally adds product 4, respectively.

Next to the fact that choosing action \( a_t \in \mathcal{A}_{s_t} \) results in a new state determined by the transition law, the user also receives a reward or cost \( r \) defined by the real-valued function \( r_t(s_t, a_t) \) for \( s_t \in S \) and \( a_t \in \mathcal{A}_{s_t} \) at time \( t \). In our example the user evaluates a decision based on three factors: time, effort, and price. For every state and action pair the cost of these resources can be specified through utility functions. These are given by the following functions:
$T(s_t, a_t) = \text{the utility of the time the user spends when taking action } a_t \text{ in state } s_t,$
$E(s_t, a_t) = \text{the utility of the effort the user spends when taking action } a_t \text{ in state } s_t,$ and
$M(s_t, a_t) = \text{the utility of the amount of money the user spends when taking action } a_t \text{ in state } s_t.$

Based on the functions $T$, $E$, and $M$ the reward function $r$ can be formulated as a weighted sum of the utilities, i.e.,

$$r_t(s_t, a_t) = \alpha_1 T(s_t, a_t) + \alpha_2 E(s_t, a_t) + \alpha_3 M(s_t, a_t), \quad (8.6)$$

where $\alpha_1, \alpha_2,$ and $\alpha_3$ are constants that denote the priority the user gives to a resource or the importance of a resource. The higher the constants the higher the user values a resource.

The description of the Markov decision problem is almost finished. We need to include the formulation of the shopping list to finalize the description. The shopping list puts constraints on the last state $s_N$ of the system. We can distinguish between two types of constraints. These constraints are due to the fact that the shopping list may contain specific items to buy (e.g., a certain product of a specific brand), and less specific items to buy (e.g., product groups).

The mathematical formulation of the first type is as follows. Let the shopping list $L$ denote the set of specific items to buy, say the list contains $n$ products, thus

$$L = \{l^1, \ldots, l^m\} \subset \{1, \ldots, M\}. \quad (8.7)$$

Then we require that $L \subset s_N$, i.e., the products of the list should be part of the contents of the shopping cart in the last state $s_N$. Since after time $t = N$ no actions follow, this constraint can be enforced by having no additional cost when $L \subset s_N$, and by adding a huge cost in the other case. Thus the cost function $r$ at time $N$ can be given by

$$r_N(s_N) = \begin{cases} 
0, & \text{if } L \subset s_N; \\
\infty, & \text{otherwise}. 
\end{cases} \quad (8.8)$$

The second type of less specific products can be formulated as well. Suppose that the user wants to buy a product from a certain product group (e.g., the user wants to buy bread, but the shopping list does not specify whether it
is a baguette, Italian bread, or a croissant). Let $\mathcal{K}$ denote the set of products that belong to that product group. Then we require that the shopping cart contains at least one product from the product group consisting of products in the set $\mathcal{K}$. Thus we require

$$\mathcal{K} \cap s_N \neq \emptyset. \quad (8.9)$$

Therefore, similar to the first case, the cost function $r$ at time $N$ can be given by

$$r_N(s_N) = \begin{cases} 0, & \text{if } \mathcal{K} \cap s_N, \\ \infty, & \text{otherwise}. \end{cases} \quad (8.10)$$

The collection of objects $(\mathcal{S}, \mathcal{A}_s, \mathcal{P}, r_t)$ refers to the Markov decision problem (see Puterman [192]) for our buying problem. Suppose that the user has a buying strategy $\pi$ that tells the user what permissible action $a_t \in \mathcal{A}_{s_t}$ to take at every time $t$ given a state $s_t$. Thus, the strategy $\pi$ can be seen as a function

$$\pi(s_t) = a_t \in \mathcal{A}_{s_t}. \quad (8.11)$$

When the user starts in state $s_0$ and adopts a strategy $\pi$, then the total cost $C^\pi(s_0)$ that he incurs is given by the total sum of all the direct costs $r_t$ at every time $t$. Thus,

$$C^\pi(s_0) = \sum_{t=0}^{N} r_t(s_t, \pi(s_t)). \quad (8.12)$$

The objective of the user is to determine a strategy such that Equation (8.12), also called the criterium function, is minimized. In our problem this means choosing the actions such that the resources time, effort, and money are optimally used. The criterium function can be minimized using a technique called dynamic programming\(^2\). The optimal strategy $\pi^*$ can be obtained through solving the optimality equations for the criterium function. The optimality equations are given by

$$C(s_t) = \min_{a_t \in \mathcal{A}_{s_t}} \left[ r_t(s_t, a_t) + \sum_{s_{t+1} \in \mathcal{S}} \mathbb{P}(s_t, a_t, s_{t+1}) C(s_{t+1}) \right], \quad (8.13)$$

for $t = 0, \ldots, N - 1$, and

$$C(s_N) = r_N(s_N). \quad (8.14)$$

\(^2\)Dynamic programming is a mathematical technique to solve sequential decision problems. Recursion is the main solution concept that appears in every formulation.
The optimality equations can be interpreted as follows. Give $C(s_t)$ the interpretation of the expected incurred cost when buying optimally when starting the system in state $s_t$. Then, an action $a_t$ yields a direct cost $r(s_t, a_t)$ and the state changes to $s_{t+1}$ according to the transition law. The expected cost that we get from thereon when continuing optimally is given by $C(s_{t+1})$. Thus, the optimality equation tells us that the expected cost $C(s_t)$ can be calculated by the sum of the direct costs of the minimizing action and the expected cost of continuing optimally in the next state.

8.4.2 User with parallel goals (trade-off): buying CDs at the Free Record Shop

In the next example we make a model using Markov decision problems\(^3\) for a user with parallel goals. There is a trade-off of values when users have parallel goals. In this example we imagine a user that is going to the Free Record Shop to buy a CD. Imagine that the user has a limited budget. The objective of the user is to buy a particular CD he likes, i.e., a CD that adds to his utility, which is modelled as pleasure here. The trade-off behavior of the user can now be modelled as follows. Assume that the Free Record Shop sells $M \in \mathbb{N}$ different CDs. Then the state space $S$ denotes all the combinations of CDs that the user can buy, and is defined by the power-set

$$S = \mathcal{P} \left( \{1, \ldots, M\} \right).$$ (8.15)

The action space $A$ is then defined by

$$A = \{1, \ldots, M\},$$ (8.16)

where action $a \in \{1, \ldots, M\}$ means that the user decides to buy CD $a$. In this example the user evaluates a decision based on two factors: the feeling (i.e., the pleasure he gets from listening to the CD) and the price of a particular CD. For every action the cost function $r(a)$ for $a \in A$ of these resources can be specified through utility functions. These are given by the following functions:

\(^3\)The theory of Markov decision problems (MDPs) studies sequential optimization of stochastic systems by controlling their transition mechanism over time. Markov decision problems are an extension of Markov Chains. In mathematics, a Markov Chain, is a discrete-time stochastic process with the Markov property. In such a process, the information at present is sufficient for predicting the future independent of the past (see Puterman [192]). The collection of objects $(S, \mathcal{P}, r)$ refers to the Markov Chain.
Mathematical formulation of search behavior

\[ U(a) = \text{the utility representing the feeling the user gets from listening to CD } a, \text{ and} \]
\[ C(a) = \text{the utility based on the price the user has to pay when buying CD } a. \]

The user wants to buy a CD which satisfies him and thus gives him the best feeling. This means that \( U(a) \) should be maximal for the user. Thus

\[ \max_{a \in A} \left[ U(a) \right]. \quad (8.17) \]

From the fact that the user has a limited budget we can gather that the price of a CD should be below or equal to the budget. Thus

\[ C(a) \leq \alpha, \quad (8.18) \]

where \( \alpha \) is the maximum amount of money the user can spend on a CD. Given these conditions we can formulate the cost function \( r(a) \) as follows:

\[ r(a) = U(a) - \gamma C(a). \quad (8.19) \]

In the function \( r(a) \), \( \gamma \) is a constant that denotes the value or importance the user gives to a CD price. The cost function \( r(a) \) in our example describes the trade-off between the user’s feeling and the price for a particular CD. Imagine that the user can take 3 actions (e.g., \( M = 3 \), thus the user can buy CD 1, 2, or 3) and that the value of the utility function is as follows for the subsequent actions:

- action 1 \( \Rightarrow U = 0, C = 0 \) (the user buys CD 1),
- action 2 \( \Rightarrow U = 1, C = 1 \) (the user buys CD 2), and
- action 3 \( \Rightarrow U = 2, C = 4 \) (the user buys CD 3).

This means that the cost function \( r(a) \) is respectively \( 0, 1 - \gamma, \) and \( 2 - 4\gamma \). We get the following figure when we draw the cost function \( r(a) \) against \( \gamma \). From Figure 8.2 we gather the following:

\[ ^4 \text{Only the line segments between coordinates } (0,2), \left( \frac{1}{3}, \frac{2}{3} \right) [\text{the intersection of the reward functions } 1 - \gamma \text{ and } 2 - 4\gamma], (1,0), \text{ and } (\gamma, 0), \text{ where } \gamma > 1, \text{ are interesting, since they represent the maximum reward for a given value of } \gamma. \]
Applications of Markov decision problems

Figure 8.2: Trade-off between the user’s feeling and the price for a particular CD.

\[
\begin{align*}
0 < \gamma &\leq \frac{1}{3} \quad \Rightarrow \text{action 3 (} U = 2, C = 4) \quad (\text{the user buys CD 3}), \\
\frac{1}{3} < \gamma &\leq 1 \quad \Rightarrow \text{action 2 (} U = 1, C = 1) \quad (\text{the user buys CD 2}), \quad \text{and} \\
1 < \gamma &\Rightarrow \text{action 1 (} U = 0, C = 0) \quad (\text{the user buys CD 1}).
\end{align*}
\]

What does this information mean? This means that when the price is not so important (very small \( \gamma \) in comparison with the value of \( \gamma \) at action 2 and 1) for the user, he will buy CD 3 even though the price for CD 3 is high (\( C \) is higher for action 3 than for action 2 or 1). The user will choose CD 3 because this CD gives him a very good feeling (\( U \) is higher for action 3 than for action 2 or 1). In this case the user’s feeling for a particular CD is more important than the CD’s price.

On the other hand when the price is very important (\( \gamma > 1 \)) for the user, he will buy CD 1 even though he does not feel much for CD 1 (\( U \) is 0 in comparison with the value of \( U \) at action 2 and 3). In this case the CD’s price is more important than the user’s feeling for a particular CD.

The user will buy CD 2 when the trade-off between the importance of the feeling and the price are more in balance for the user (\( \gamma \) is not too small, and \( U \) and \( C \) are not so small or big in comparison with the values of \( U \) and \( C \) at action 1 and 3). In Figure 8.3 we have visualized the areas in which the user decides to buy a particular CD.

In this example we have modelled a situation in which a user has to choose between several products. To evaluate each product the user has
several criteria modelled as utility functions. Since every product leads to different evaluations on the different criteria, there is a trade-off to be made here. In this example we have shown that the importance the user assigns to a criteria determines his behavior. Therefore, for general problems, it is important to know what criteria the user has and how important they are. Empirical studies can reveal these data.

8.4.3 Look and click behavior of web visitors

In the next example we provide a model to study click behavior of web visitors, and consequently where they are looking. This model is quite flexible and allows us to study different settings. First, we can study a setting that can be used to decide how to aid a web visitor in achieving its search goal by placing navigation aid(s) on the website. Second, we can also study how web visitors can be triggered to change their goal. Finally, we will show how the model can be used to describe the web visitor’s wild search behavior. We start with the description of the first setting.

Navigation aids

In this example we imagine the following site map of a website. In Fig-

---

Figure 8.3: Visualization of the areas in which the user decides to buy a particular CD.
Figure 8.4: Example of a site map of a website.

The nodes represent the webpages and the lines the links between the webpages. The **hyperlink structure** of the site map can be given by

\[
E = \{(S_0, S_1(1)), (S_0, S_1(2)), (S_0, S_1(3)), (S_1(1), S_2(1)), \ldots\}. \tag{8.20}
\]

The state space, \(S\), denotes the *collection of webpages* and is defined by

\[
S = \{S_0, S_1(1), S_1(2), \ldots, S_3(9), S_3(10)\}. \tag{8.21}
\]

We assume that the goal of the web visitor is to reach page \(S_3(1)\). This can be realized by taking actions in the system (which correspond to clicking on a link). The action space \(A_s\) when the system state is \(s \in S\), i.e., the web visitor is viewing webpage \(s\), is given by

\[
A_s = \{a \mid (s, a) \in E\} \quad \forall \ s \in S \setminus \{S_3(1), \ldots, S_3(10)\}. \tag{8.22}
\]

Here, \(A_s\) denotes the *set of webpages* where the web visitor can go starting from page \(s\), the webpage which is shown on the web visitor’s display. The action \(a \in A_s\), means that the web visitor decides to go to a new webpage. Note that when the web visitor reaches a final state, i.e., states \(S_3(1), \ldots, S_3(10)\), then no action is possible. We assume that in these states,
no further actions are taken. The action space can also be written as

\[ A_s = \begin{cases} 
\{S_1(1), S_1(2), S_1(3)\} & \text{if } s = S_0, \\
\{S_2(1), S_2(3)\} & \text{if } s = S_1(1), \\
\{S_2(2), S_2(4), S_2(5)\} & \text{if } s = S_1(2), \\
\{S_2(2), S_2(3), S_2(6), S_2(7)\} & \text{if } s = S_1(3), \\
\ldots
\end{cases} \tag{8.23} \]

In this example we made two assumptions:

1. The web visitor is not clicking on a link to go back to a previously visited page, and

2. The web visitor has to click on a link to go to a new page.

Since the current state (i.e., the current webpage) is known with certainty the transition law \( P \) in this example is deterministic, i.e., \( P \) is either 0 or 1. Thus,

\[ P(s, a, s') = \begin{cases} 
1 & \text{if } s' = a \text{ and } (s, a) \in E, \\
0 & \text{otherwise},
\end{cases} \tag{8.24} \]

where \( s \in S \) is the current state (i.e., the current webpage on the web visitor’s display), \( a \in A_s \) is the decision the web visitor takes to go to a new page, and \( s' \in S \) is the new state (i.e., the new webpage on the web visitor’s display) after the decision has been taken.

The reward function \( r(s, a) \in [0, 1] \) gives the web visitor perception for the relevance of a link or webpage. If the web visitor’s goal is to navigate to \( S_3(1) \), the relevance for this page will be high for the web visitor. Then the reward function might be \( r(s, a) = 1 \) if \( a = S_3(1) \) and \( r(s, a) = 0 \) for all \( a \neq S_3(1) \) given that \( (s, a) \in E \). This means that when reasoning from the web visitor’s perception the probability, \( P \), to click on a link depends on the relevance for a link or webpage for the web visitor.

We first consider a situation in which the web visitor cannot infer any information regarding on which link to click to reach \( S_3(1) \). Therefore, this situation corresponds to a web visitor clicking randomly on one of the links presented on its current page. Hence,

\[ P(s, s') = \begin{cases} 
\frac{1}{n_s} & \text{if } (s, s') \in E, \\
0 & \text{otherwise},
\end{cases} \tag{8.25} \]
with \( s, s' \in S \), where \( n_s \) is the number of links on webpage \( s \). Therefore the probability of consecutively visiting webpages \( s_0, s_1, \ldots, s_k \) is given by
\[
P(s_0, \ldots, s_k) = P(s_0, s_1) \times \cdots \times P(s_{k-1}, s_k).
\]
The probability to go from \( S_0 \) to a page on \( S_1 \) is given by
\[
P(S_0, S_1(j)) = (A_0)_j. \tag{8.26}
\]
The probability to go from a page on \( S_1 \) to a page on \( S_2 \) is given by
\[
P(S_1(i), S_2(j)) = (A_1)_{ij}. \tag{8.27}
\]
Likewise, the probability to go from a page on \( S_2 \) to a page on \( S_3 \) is given by
\[
P(S_2(i), S_3(j)) = (A_2)_{ij}. \tag{8.28}
\]
In Equations (8.26), (8.27), and (8.28), \((A_0), (A_1), \) and \((A_2)\) are matrices, which are given below.

\[
A_0 = \begin{pmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3}
\end{pmatrix}, \tag{8.29}
\]
\[
A_1 = \begin{pmatrix}
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\
0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4}
\end{pmatrix}, \tag{8.30}
\]
\[
A_2 = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{6} & 0 & 0 & \frac{1}{6} & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}. \tag{8.31}
\]

With these matrices we can calculate the probability to go from \( S_0 \) to a page on \( S_3 \) by calculating \( A_0 \times A_1 \times A_2 \), which results in
\[
P_{S_0 \rightarrow S_3} = \begin{pmatrix}
\frac{13}{72} & \frac{13}{72} & \frac{35}{216} & \frac{11}{216} & \frac{7}{72} & \frac{17}{216} & \frac{17}{72} & \frac{17}{72} & \frac{1}{72} & \frac{1}{72} & \frac{1}{72}
\end{pmatrix}, \tag{8.32}
\]
where \( P_{S_0 \rightarrow S_3} \) is the probability from \( S_0 \) to a page on \( S_3 \).

The calculation above shows that the probability to reach page \( S_3(1) \) is \( \frac{13}{72} \approx 0.18 \). This is quite a low probability to realize the web visitor’s goal. Based on this information one can design navigation support to aid the web
visitor in clicking the right link (the navigation support can be designed with the help of the transition diagrams of Section 7.5). For example, suppose that the navigation support on $S_0$ increases the probability that the web visitor will click on the link $(S_0, S_1(1))$. Assume the probability increases from $\frac{1}{3}$ to 1 (thus the web visitor clicks with certainty on $S_1(1)$). Then the probability to arrive on a page at $S_3$ starting from $S_0$ is

$$P_{S_0 \rightarrow S_1(1) \rightarrow S_3} = [1, 0, 0] \times A_1 \times A_2 = \left( \begin{array}{ccc} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \right).$$

From Equation 8.33 we can conclude that the web visitor will visit pages $S_3(1)$ with a probability of respectively $\frac{1}{2}$. Hence, the navigation support has increased the probability to realize the web visitor’s goal. Similarly, when the probability increases to $\frac{3}{4}$, the probability to reach $S_3(1)$ starting from $S_0$ is $\frac{73}{192} \approx 0.38$. Consequently, the model that we have described can be used to determine how the navigation support can be designed to support web visitors in achieving their goal.

**Goal changes**

In the example we have seen how the model can support the design of navigation support. However, the example can also be used to influence the goal of web visitors. Consequently web visitors may change their goals. The increase in probability can then be seen as a state trigger (e.g., an advertisement, price, logo, or color) to influence the web visitor’s goal and hence his search behavior. Thus, a web visitor starts at page $S_0$ and is presented with a state trigger that increases his probability to click on $S_1(1)$. Consequently, the probability to reach $S_3(1)$ increases as well.

**Wild search**

Note that the probabilities to reach pages within $S_1$, $S_2$, or $S_3$ describe the wild search behavior of a web visitor. The more vague the search, the more uniform the probability distribution is, e.g., $P_{S_0 \rightarrow S_1} = \left( \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right)$. The more precise the search is (the case in which a web visitor has a precise goal), the more the distribution tends to select one outcome with certainty, e.g., $P_{S_0 \rightarrow S_1} = (1, 0, 0)$. The distributions in between reflect the continuum of stages of wild search, such as a web visitor having parallel goals or a web visitor having a hedonic search strategy.
Feasibility of the approach

The Markov decision problem presented in the subsection is a natural way to model click behavior of web visitors. Although, the modeling approach seems natural, i.e., taking the webpages as states and the links between them as possible transitions, it is formulated from the viewpoint of a web designer. Hence, in practice, it is very difficult to make the reward function \( r(s,a) \) concrete, since it is unknown what the web visitor is searching for. Instead, when a modeling approach is used which is formulated from the viewpoint of the web visitor, this problem disappears. Then it is possible to differentiate between groups of web visitors with different search behavior.

In the next chapter, we study examples of website optimization that are used in practice. However, in the discussion we will show that all these methods do not differentiate between different user groups. Therefore, we will develop a model that addresses this problem in the chapter thereafter.

8.5 Conclusion

Search behavior is a complex process that is difficult to study. In order to systematically study search behavior, there is a need for models that quantify features of search behavior. Markov decision problems provide a framework in which this can be established. Therefore, we have introduced the basic components of a Markov decision problem in this chapter. We modelled the user’s search behavior in the framework of Markov decision problems, which we applied in both “non-web” and web situations. We have developed a model for the click behavior of web visitors, which was applied to the category navigation aids. We then discussed the applicability of the model to different categories (e.g., precise goals, goal changes, parallel goals, hedonic search strategy), which we used in Chapter 4 and 5. We also observed that this model also describes the wild search behavior of a web visitor.

The examples in this chapter are meant to familiarize oneself with Markov decision models. In the Chapter 10 we develop a model for dynamic website optimization through autonomous management of design patterns based on the concepts developed in the chapter. The observation results of Chapter 5 are used to estimate model parameters. Then the results of the estimate are analyzed to optimize websites.
Mathematical formulation of search behavior
Chapter 9

Techniques and tools for website optimization

9.1 Introduction

In Chapter 8 we introduced the basic components of a Markov decision problem. We discussed mathematical and notational subtleties in order to formulate search behavior in this framework. In particular, we modeled look and click behavior of web visitors. At the same time, we identified that there are some practical issues to overcome before that model can be applied in practice for website optimization. In this chapter, we provide an overview of some techniques that are applied in practice for website optimization. We can categorize these techniques into categories, which are described in Section 9.2. In Section 9.3 we refer to some concrete examples of tools that use these techniques to optimize websites. We end the chapter with a discussion on the applicability of the techniques in Section 9.4.

9.2 Techniques to optimize websites

There are many commercial parties that offer tools for optimizing websites. However, the techniques that are used by these parties do not differ that much from each other. In fact, one can identify two categories of optimization techniques that are used in practice.

The first technique focuses on web analytics, see, e.g., [64, 161, 226, 23]. The tools that adopt these techniques analyze the logfiles of the webserver
and gather various other statistics to answer questions on the performance of the website and provide insight into the behavior of web visitors. For example, typical questions that can be answered are ‘How many unique web visitors per month are attracted by my website?’, ‘How popular is my website?’, ‘Where do my web visitors come from?’, ‘Which browsers do my web visitors use?’, ‘Which keywords are used to land on my website?’ It is also possible to analyze the behavior of the web visitor as he visits the website. This analysis is carried out by placing the popularity of each section of the website in different activity maps (e.g., click maps, mouse move maps, scroll reach maps, and attention maps). In addition, statistics are also collected by recording all actions and mouse movements of the web visitor as a movie that can be watched back (an example of such a system is [64]). The first technique mainly provides a passive manner on the collected statistics which have to be evaluated by a web designer to decide on adjustments to improve the website.

The second technique deals with tools for website testing and optimization. In this setting, several versions of a website are created. Then, the different versions are tested on their performance by presenting them to the web visitors. The methods to test these versions are A/B testing and Multivariate Testing (MVT). A/B testing (used by, e.g., [180, 206, 232, 181]) is used to test different versions of an entire website, whereas MVT is used when different components of the website are varied to determine the best combination. Since these two techniques are quite relevant to our research, we elaborate more on A/B testing in Section 9.2.1, and on multivariate testing in Section 9.2.2.

### 9.2.1 A/B testing

A/B testing (also known as split testing or bucket testing [245, 235]) is the simplest model of testing. It is defined as testing two (or more, the so-called A/B/n testing) different versions, say A and B, of the same variable (e.g., a green theme vs. a red theme). A/B testing is not designed to test multiple variables simultaneously. The influence of the different variables on the outcome cannot be determined and the interaction between the variables can also not be measured.

A/B testing can be done consecutively or simultaneously. In consecutive testing, one tests the performance of version A first, and then tests the performance of version B. When you test consecutively, it is possible that
the test results are influenced by time effects. It is therefore better to test simultaneously. In simultaneous testing, one mixes both versions to assess the performance of both versions.

The main purpose of A/B testing is to improve conversion rates (the percentage of web visitors converted to customers) of, for example, websites. A/B testing clearly shows the effect of different versions. When you test a new page design compared to the existing page design, then you can test whether the new page design scores better and how much better than the existing page design. One advantage of A/B testing is that web visitors do not know that they participate in an experiment. Web visitors are in their natural setting and show natural behavior that is not influenced by their awareness of being involved in an experiment.

A/B testing schemes are easy to set up and the results are easy to interpret. They often provide quick results, because there are in total less variations or combinations compared to MVT (see also Section 9.2.2). A/B testing can therefore be used if you want quick results. Although A/B tests provide quick results for one variable or an entire webpage, MVT is quicker and more accurate when you are testing multiple variables. Even if you have a webpage that generates few web visitors, an A/B test can sometimes be better than a MVT. This is because MVT forms more combinations. With a small number of web visitors the duration of the test would be too long in order to obtain reliable results.

A/B testing is also suitable for testing the layout (e.g., logo and navigation positions), the look and feel (e.g., images and colors), and the content of a webpage.

### 9.2.2 Multivariate Testing (MVT)

In contrast to A/B testing where only one element on a webpage at a time is varied, multivariate testing considers various multiple different elements at the same time. In multivariate testing, different sections or elements on a webpage are identified which affect the conversion rate. Different variations of those elements are created, which are then combined to give rise to multiple different versions of the website.

Multivariate tests take more time than A/B tests to show results, but are more likely to produce better results. MVT not only tests which elements affect the conversion rate and which ones do not, but it also is able to identify the optimal combination of page elements that achieves the highest response
rate. Note, however, that the method is considerably more complicated than A/B testing because the different elements are correlated, i.e., two elements together may have a strong effect on the conversion rate, whereas a single element alone may not. This is a difficult problem to resolve, and in practice many websites use A/B/n testing on the collection of versions without taking the correlated effects into account. This creates the net effect of doing many simple experiments at the same time.

9.3 Examples of some tools to optimize websites

In Section 9.2 we discussed two categories of techniques. Examples of tools that belong to the realm of web analytics are Clicktale [64], Mouseflow [161], Userfly [226], and Google-analytics [23]. These tools try to optimize websites by analyzing website statistics generated by web visitors. In Section 9.3.1 we elaborate on some of these tools in greater detail.

Google Website Optimizer [180], SiteSpect [206], Visual Website Optimizer [181], and Vertster [232] belong to the second technique of website testing and optimization. These tools use A/B testing and MVT to select the best version from a given number of website versions that are prepared in advanced. We discuss some of these tools in Section 9.3.2

9.3.1 Web analytics tools

Figure 9.1 shows an example of website statistics that is presented by the Google Analytics tool. The figure shows a part of all the collected website statistics. In the Site Usage part of the website statistics we see, e.g., the number of web visitors (Visits) that visited this particular website, the number of webpages that are visited (Pageviews), and the percentage of web visitors that left the website after visiting the first webpage (Bounce rate). The website statistics give insight into the demographics of the web visitors, i.e., from which part of the world are web visitors browsing from (Map Overlay). This Map Overlay can be zoomed in to city level. The Traffic Sources Overview provides information of how the web visitor reached the website (e.g., by search engines or by typing the URL direct in the webbrowser). In the Content Overview we can see how many times a webpage was viewed.

Clicktale [64], Mouseflow [161], and Userfly [226] all work in the same way, where Clicktale is more comprehensive than the others. Therefore, we will highlight some features of Clicktale here. Clicktale uses, among other
Examples of some tools to optimize websites

Figure 9.1: An example of website statistics that is presented by the Google Analytics tool.

features, heatmaps (see Figure 9.2) to collect web visitor data. With click heatmap, for example, you can see where the web visitor clicked on the webpage. By aggregating the mouse movements of thousands of visitors on a webpage, Clicktale creates a visual representation (a mouse move heatmap) of what visitors are looking at and focusing on within the page. The attention...
202 Techniques and tools for website optimization

Figure 9.2: Example of three types of heatmaps: the attention heatmap in the upper left corner, the click heatmap in the lower left corner, and the mouse move heatmap on the right.

*Action heat map* is an extension of the mouse move heatmap and is in particular applicable to long parts in websites. This heatmap shows how far web visitors scroll down the webpage and on which parts of the webpage they stay the longest. Besides heatmaps, Clicktale can record mouse clicks, mouse movements, keystrokes, and scroll actions of a web visitor as a movie to see the web visitor’s activities on the website.

### 9.3.2 Website testers and optimizers

Google Website Optimizer [180], SiteSpect [206], Visual Website Optimizer [181], and Vertster [232] use A/B testing and MVT test methods to select the best version from a given number of website versions, which are prepared in advance. Before testing the various versions of a website, one should define various hypotheses on which elements on the website might be successful. This could be, e.g., different backgrounds, headers or design patterns. The tests can start if the various versions have been created. Google Website
Examples of some tools to optimize websites

Optimizer, e.g., tests the different versions simultaneously. In this way the influence of external factors and seasonal variations will be equal for all website versions. To test all the versions simultaneously, Google Website Optimizer splits all the traffic to the website. This means that not every web visitor gets the same version of the website to see. A cookie takes care that the web visitor always gets to see the same version once he has been presented a website version (conversion can take place after multiple visits). If the tests have run long enough then the results will look like Figure 9.3. There is a graphic area in the figure that shows how much better or worse a particular variation is performing above the original control version, the baseline without any alterations. The following colors can occur in the graph:

- Red: It is certain that this combination does not perform well.

- Yellow: This combination performs a little better or worse than the original, but it is still uncertain.

- Green: It is certain that this combination performs well.

- Gray: This combination performs as good as the original.

<table>
<thead>
<tr>
<th>Combination</th>
<th>Estimated Conversion Rate Range [?]</th>
<th>Chance to Beat Cnt. [?]</th>
<th>Chance to Beat Alt. [?]</th>
<th>Observed Improvement [?]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>17.2% ± 6.4%</td>
<td>0.04%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combination 1</td>
<td>30.9% ± 13%</td>
<td>89.3%</td>
<td>37.0%</td>
<td>79.3%</td>
</tr>
<tr>
<td>Combination 2</td>
<td>29.5% ± 12%</td>
<td>87.8%</td>
<td>25.3%</td>
<td>71.1%</td>
</tr>
<tr>
<td>Combination 3</td>
<td>28.8% ± 10%</td>
<td>88.4%</td>
<td>19.2%</td>
<td>66.8%</td>
</tr>
<tr>
<td>Combination 4</td>
<td>26.7% ± 8.5%</td>
<td>84.8%</td>
<td>8.88%</td>
<td>54.7%</td>
</tr>
<tr>
<td>Combination 5</td>
<td>24.4% ± 10%</td>
<td>77.9%</td>
<td>4.21%</td>
<td>41.3%</td>
</tr>
<tr>
<td>Combination 6</td>
<td>22.4% ± 10%</td>
<td>70.3%</td>
<td>2.25%</td>
<td>29.9%</td>
</tr>
<tr>
<td>Combination 7</td>
<td>21.6% ± 8.8%</td>
<td>68.1%</td>
<td>1.30%</td>
<td>25.4%</td>
</tr>
<tr>
<td>Combination 8</td>
<td>20.7% ± 8.2%</td>
<td>65.1%</td>
<td>0.60%</td>
<td>20.2%</td>
</tr>
<tr>
<td>Combination 9</td>
<td>19.5% ± 6.0%</td>
<td>60.1%</td>
<td>0.31%</td>
<td>13.2%</td>
</tr>
<tr>
<td>Combination 10</td>
<td>15.1% ± 8.7%</td>
<td>40.2%</td>
<td>0.01%</td>
<td>-12.6%</td>
</tr>
<tr>
<td>Combination 11</td>
<td>11.2% ± 7.0%</td>
<td>22.3%</td>
<td>0.00%</td>
<td>-34.8%</td>
</tr>
</tbody>
</table>
9.4 Conclusion

In this chapter we have explored several techniques for website optimization in practice. A first category of techniques is rather passive and analyzes web logs and user behavior. These statistics are relevant to obtain insight into how a user came about to visit the website and what he does during his visit. However, the analysis requires that a web designer looks and interprets the data and then devises an improvement to the website.

A second category of techniques is focused on website testing and optimization. In principle, variations on A/B testing and MVT are used. The main question that is raised when using these techniques is when one should stop testing (in, e.g., Google Website Optimizer and similar systems). Moreover, a website version is not chosen autonomously. Furthermore, the moment at which a website version should be chosen is not always the same and has to be determined by the web designer. These shortcomings are rather prohibiting in deployment of these techniques in practice for obtaining the full benefit of website optimization. Therefore, there is a need for new models that resolve these issues. In the next chapter, we present our solution to overcome these problems.