21

Rationality

Why do some users prefer to visually search a menu for a command while other users prefer to use keyboard shortcuts? Why do users avoid looking at advertisements on web pages but are attracted to large graphics and titles such as logos? Why do some users type with two thumbs in one context but only one finger in some other context? Why do users often reduce their walking speed when typing on a mobile device? Why do users avoid cognitively demanding tasks like generating new passwords? These questions are instances of a foundational question in human–computer interaction (HCI) research: Why does some particular behavior emerge in a given interaction setting?

This chapter introduces the notion of *rationality*. This allows us to provide explanations for interactive behavior due to users attempting to make the most out of the choices available to them.

To state that a user's choice is *rational* means that it is selected with the expectation that it yields the highest *utility* out of the available options. Thus, the notion of rationality does not only explain what a user does; it also considers what that user *could* have done but chose not to.

The concept of rationality has its roots in economics, where it was developed to study how people *should* act in economic decision-making. In such settings, the idea is that people reach their goal, such as maximizing their return, by maximizing utility. However, empirical studies demonstrated that people do not behave as such a *normative* view suggests. Instead, people constantly fail to consider all options, estimate the value of the choices provided to them, and choose the economically optimal option.

The present understanding is that users are not fully rational. The ability to make rational decisions is limited by both internal abilities, such as those posed by perception and cognition, and by the structure of the environment in which one operates. Therefore, to predict user behavior, it is insufficient to merely understand the rewards of interaction: the goals and utility. We must also understand users' limits, which are usually called *bounds* in this context.

As an example, consider the following situation: "I promise to give you money based on your preferred option. Which one would you choose?"

1. I give you £50 for sure.
2. I give you £100 with a 50% probability.

From an economic theory perspective, these two options have equal outcomes. The choice does not matter because the expected return is the same. Yet, some respondents might consistently prefer options like the first one, demonstrating *risk awareness*. By contrast, *risk-seeking* people tend to prefer the second option.

In HCI, there are many reasons for behaving in a seemingly irrational way. A user may fail to choose the (normatively) best option, for example, due to constraints such as time pressure, limited knowledge, or an incomplete set of beliefs. They may, for example, not know what the options are or what their attributes are. They may also have preferences for certain outcomes. For example, they may want to factor in the level of risk involved. They can prefer to obtain rewards

Introduction to Human-Computer Interaction. Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta, Oxford University Press. © Kasper Hornbæk, Per Ola Kristensson, and Antti Oulasvirta (2025).
DOI: 10.1093/oso/9780192864543.003.0021

quickly or be more patient and work longer for them. Thus, contrary to a common misconception, rationality does not imply the user is all-knowing, all-capable, or able to achieve what is objectively the best outcome for them. Users can only behave as well as they can.

The descriptive view of rationality explains people's actual behavior when making choices. Descriptive theories attempt to capture causes behind behavior that, from a normative perspective, may appear irrational. This view enables predictions of user behavior in real-world circumstances.

In particular, *bounded rationality* states that we are only rational to the extent allowed by the involved constraints, or bounds. The term *satisficing* is used to describe how users tend to behave when facing a complex decision-making problem. It refers to settling on a satisfactory but not optimal solution in the normative sense. Instead of searching for the very best option, users tend to search only as long as it takes them to identify a satisfactory option. Here are a few examples of satisficing in HCI:

- When looking through search results, users rarely look beyond the first page; they often settle on something that appears close to the top.
- Instead of investing effort into learning shortcut commands, such as Ctrl-C for copy, users often just look for the option in a drop-down menu.

How do users determine the satisfactory level? They use a *reference point*, for example, a previous experience. If that point is low, they may be satisfied with a much lower experience than if the point is high. Consider looking for sneakers in a web shop that offers thousands of them. After a quick scan, you bump into a pair of good-looking shoes offered for £ 40. Do you keep looking or stop now and buy the pair of shoes you just found? Your behavior would likely depend on your reference point, for example:

1. You believe that the kind of shoes you love should be available for £ 80.
2. You believe that the kind of shoes you love should be available for £ 40.

It is rational to satisfice: In most cases, good enough is good enough. Satisficing is also a good strategy for an agent with a limited time budget or an unwillingness to invest cognitive effort. For example, a user choosing a way to travel through a web service may not need to analyze all the options and fully understand their costs and benefits; they may choose the first one that is deemed good enough. In general, finding the normatively optimal solution to a *multi-attribute choice problem*, a problem where a user needs to evaluate and compare many attributes, requires extensive pairwise comparisons and retaining a lot of information in memory. We can avoid that effort if we find even just one option that is good enough. Satisficing is therefore an important concept because it factors in the amount of effort involved in finding optimal solutions.

The concept of rationality views users as agents. An *agent* is an actor with the ability to choose actions in pursuit of some goal or reward. An agent's behavior is understood in terms of the agent making a *choice*: From numerous possible ways to act, the agent chooses and carries out a specific course of action. This is a general concept in HCI since even the act of pressing a button can be understood as a choice. For example, a user pressing a button can vary the amount of force and the timing of the press in different ways. This example illustrates that for behavior to be analyzed as choice, we do not need to assume the user is consciously aware of the options available to them. It is possible to analyze even relatively low-level behavior, such as eye–hand coordination in computer use, through the lens of rationality.

To model users as agents, we need to assume they tend to choose actions that maximize their *expected utility*. In other words, users do what they believe is best for them. For an HCI researcher wanting to apply these theories, five basic concepts must be understood:

States: States describe the possible states that the user may enter.

Actions: In each state, the user can choose from several options; these options are called actions.

Rewards: The rewards represent what the user wants and are associated with different states.

Costs: Costs are negative rewards that the user incurs for transitioning between states or for being in states that are not good for them.

Environment: The space of all states, actions, and rewards is called the environment.

In any given *state* of the environment, a set of *actions* is possible. States can be specified in different ways in theories of rationality. They can be specified by the user interface, the task context, or the user's beliefs. Certain states are associated with *rewards* that are valuable for the user, such as information that is interesting or getting an email sent to a colleague. However, some states or transitions incur *costs*, such as the cost of spending time or energy. Such costs are *negative* rewards. Costs can be quantified in many ways, such as the time or effort invested.

Utility refers to the agent's consideration of positive and negative rewards when deciding how to act. Contrary to the received view from economics, in HCI, utility is often not related to money or other externally defined values. Instead, it is mostly defined by reference to *intrinsic* rewards, such as being productive, experiencing enjoyment, or being appreciated by others; see the discussion of needs and motivations in Chapter 6. Sometimes, intrinsic rewards are closely associated with extrinsic states, such as pressing a particular button to send an email or receiving a particular message.

The rest of this chapter reviews four theories of rationality relevant to HCI and illustrates them with examples. Table 21.1 compares these four theories. They share a focus on the emergence of interactive behavior; in other words, they predict how users choose to behave in certain given circumstances.

Economic models: These models predict that users choose actions by considering the costs and rewards of those actions. For example, users do not form maximally informative search queries when using search engines. Their queries' lengths can be predicted by considering

Table 21.1 Four theories of interactive behavior.

Theory	Main idea
Economic models	Users choose to interact in a way that maximizes the payoff, that is, the ratio of the benefits to the costs of actions.
Rational analysis	User behavior adapts to the ecology, that is, the experienced distribution of rewards in the environment.
Information foraging theory	Users are "informavores" who navigate information environments, such as a web site, in a way that attempts to maximize information gain while factoring in the time costs of actions.
Computational rationality	Users interact in a manner that attempts to maximize an expected future reward; however, such attempts are bounded by the capabilities of their bodies, their cognition, and the environment.

the effort and time needed to generate them (i.e., the costs) versus the additional gain from each added term [30].

Rational analysis: Such analysis is based on the assumption that the way users experience rewards being distributed in the environment shapes their behavior. This distribution is called the *ecology*. For example, when users observe an interface for the first time, they do not look at it randomly. Instead, they tend to look at those regions that they expect to contain the most information. Such gaze tendencies are an example of behavior adapted to the interface ecology.

Information foraging theory: This theory views users as "informavores" to predict how they search—*forage*—for information. It assumes that at any given time, users choose where to go next by estimating the information gain and the time cost for each option. For example, when searching for a product in a web store and looking at its menu, which link will a user click? According to this theory, the user chooses the link with the highest ratio of expected relevance—or *information scent*—to the navigation cost, such as the time spent navigating the site [661]. A study of large-scale web-clicking data employed this theory to explain why certain distributions of web page hits emerge on web sites. Huberman et al. [362] proposed a mathematical model that assumes that at any page, users decide to continue clicking as long as its information scent exceeds some threshold. This information scent can be computed using information foraging theory (IFT).

Computational rationality: This concept assumes that in computer use, we face a sequential decision-making problem: We must choose which of a set of immediate actions might eventually get us to the state we care about. To do this, we estimate the values of achievable actions. However, in doing this, we are limited by our beliefs, bodies, and cognitive capabilities. The sequential decision-making problem is computationally hard, and machine learning methods are required to predict users' actions in concrete task environments. Recent examples include generative user models for typing, multitasking, menu use, and driving tasks [632].

These four theories differ in the factors they include and how the agent's decision-making problem is formulated. As such, the theories differ in how easily they help us find a solution to the user's decision-making problem. For example, while economic models can be analyzed via pen-and-paper calculations, computational rationality requires a machine learning approach to arrive at a solution.

21.1 Economic models

Economic models examine user choices in terms of *payoffs*. A payoff refers to the benefits that are left after the costs have been subtracted. When the costs are high, the payoff may be low. When the benefits are high and the costs are low, the payoff is high. In economic models, a rational user chooses actions that maximize the payoff.

In traditional economic models, benefits and costs refer to financial variables, such as money; in HCI, they refer to subjectively relevant variables that can be positive or negative, such as the expenditure of time (a cost) and the gain of interesting information (a benefit).

Applications of economic models in HCI start by analyzing the payoffs of each option. These are described in terms of information or efficiency (the benefit) against a variable such as time (the cost). Then, a *design variable* is defined, such as the length of a query a user is typing. The payoff is then computed as a function of this design variable. The shape of this function helps us

identify sweet spots—values of the design variable where the payoff is high or reaches its peak. Paper Example 21.1.1 illustrates a worked example of payoff calculations for search query lengths.

Paper Example 21.1.1: Economic modeling of query-based search

To illustrate the computation of payoffs, let us consider the example of search queries [30]. The question is how long a search query a user should type, given that entering a longer query has a higher time cost but may yield a benefit in the form of better results. We start by looking at the time cost of entering a query of varying length and then look at its effect on the quality of the information obtained from the search results. A rational user chooses the optimal query length according to the cost–benefit trade-off.

To apply the economic model to search engine use [30], we make two assumptions. First, let us assume that the time cost c of entering a query is linearly related to the number of words W:

$$c(W) = W \cdot c_w, \tag{21.1}$$

where c_w is the time spent to enter one word. Simply put: The more words you type, the longer it takes.

Second, let us assume a benefit function $b(W)$ that describes the accumulated information in search results as W increases. Let us further assume that the search results are presented in some relevance-based order. This implies that $b(W)$ yields diminishing benefit as W increases:

$$b(W) = k \cdot \log_a(W + 1), \tag{21.2}$$

where k and a are empirical scaling factors. Simply put: As a increases, additional words in W contribute less to the total benefit.

To compute the payoff (also referred to as profit in the figure below) π, we need to combine these two terms as follows:

$$\pi = b(W) - c(W) = k \cdot \log_a(W + 1) - W \cdot c_w. \tag{21.3}$$

To find the optimal query length W^*, Equation 21.3 is first differentiated with respect to W:

$$\frac{\partial \pi}{\partial W} = \frac{k}{\log a} \cdot \frac{1}{W + 1} - c_w = 0. \tag{21.4}$$

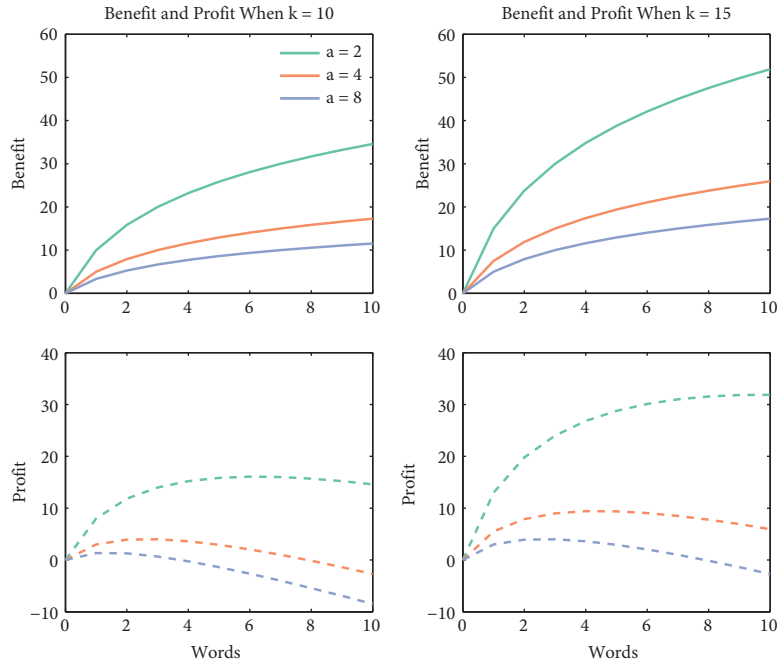
This can be solved for W^* as follows:

$$W^* = \frac{k}{c_w \cdot \log a} - 1. \tag{21.5}$$

The figure in this paper example box shows the model with different values of the parameters a and k. For example, when $k = 10$, the profit peaks at around 1–4 words; this result corresponds well with empirical data on query lengths.

continued

Paper Example 21.1.1: Economic modeling of query-based search *(continued)*



21.2 Rational analysis

Consider an ant walking on a beach. How much of the ant's route is determined by the ant and how much of it is determined by the environment? The small bumps and valleys in the sand are somewhat predictive of the ant's path.

An ant on a beach illustrates the paradox that simple organisms can produce complex behaviors. A simple control rule, when immersed in a complex environment, can produce behavior that appears complex. By implication, knowing someone's environment can be very useful for inferring their behavior.

To see the connection to HCI, instead of an ant, think about a user navigating in a graphical interface. How much is a cursor's movement controlled by the user versus the design of the layout? For example, if you move your cursor in a menu, the columns and rows of the menu are highly predictive of how your cursor will move. Moreover, the first commands of a menu are more likely to be selected than those that are hidden deeper within the menu structure. Additionally, a user's gaze and mouse cursor are less likely to dwell on areas that contain little information, such as white space. Rational analysis generalizes this observation into a principle of rational behavior.

Rational analysis is a theory of rational behavior proposed by Anderson and Schooler [21]. It examines the distribution of rewards in the environment to explain how users adapt their behavior. According to rational analysis, behavior is sensitive to the statistical distribution of rewards in the environment that a user has experienced. Users learn the way rewards are distributed through continued exposure to an environment and adapt their behavior accordingly. A user's behavior is rational because it is tuned to the distribution of rewards in the environment—the ecology.

The theory can be used to predict user behavior in cases where (1) the reward distribution can be charted and (2) when users have had enough time to adapt their behavior.

For example, consider the spatial distribution of rewards on a web page. Some elements, such as decorative elements, contain virtually no information. Other elements, such as text and titles, typically contain a lot of information. Graphics can be either informative, such as a logo, or non-informative, such as a decorative background image. When the user encounters a web page for the first time, where should they look? Rational analysis predicts that users adapt their behavior to the distribution of informative and non-informative elements. For example, the logo of the site owner is typically at the top of the web page, while a navigation menu is commonly presented either on the left or at the top of the page.

Rational analysis predicts that the probability of a user gazing at a location is proportional to the expected location of information across previously encountered user interfaces. For example, what a user initially gazes on in a mobile app depends on where the interesting elements statistically reside. If a designer wants to predict this, they simply need to know how other apps that the user might use are designed.

Visual statistical learning is a research topic in perception that studies how the statistical distribution of our environments affects the deployment of gaze. The history of interface designs encountered by users is a valuable source of information for a designer. For instance, users can find elements faster if they are placed at statistically expected locations. For example, users who look at mobile apps typically focus on the top-left quadrant (Figure 21.1). Paper Example Box 12.2.1 provides another example of rational analysis.

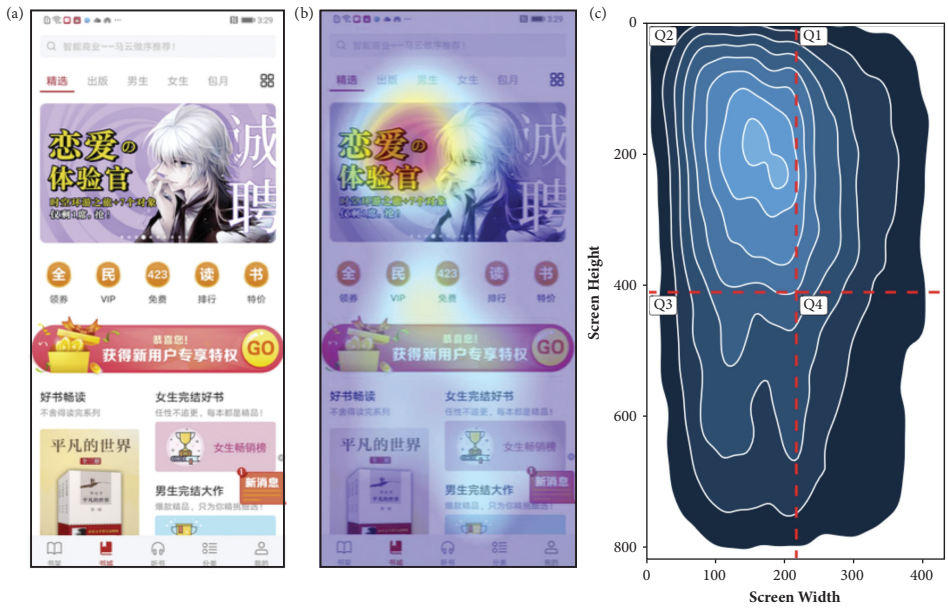


Figure 21.1 The gaze pattern of a user looking at a mobile user interface is biased toward the top-left quadrant. The figure shows (a) the original user interface, (b) the aggregated gaze pattern when viewing the content freely, and (c) the aggregated pattern over *all* mobile user interfaces in a study [472]. According to rational analysis, this phenomenon arises due to the adaptation of gaze behavior to the distribution of information in the ecology of mobile user interfaces.

Paper Example 21.2.1: Forgetting passwords is rational

Reusing a password is unwise and opens those accounts to attacks. As a consequence, most users have several passwords that need to be remembered for various systems in use. However, the more passwords we have, the harder it gets to create them and recall them when they are required.

To recall a password, we need to retrieve it from long-term memory. However, what determines how easily it can be retrieved? Rational analysis suggests an answer to this question. It assumes that human long-term memory evolved to help survival by anticipating organismically important events. It is evolutionarily important to remember things that are important for survival. Therefore, the expected value of remembering a thing in the future should affect the probability of recalling it.

Since most memory usage is not directly related to survival, Anderson and Schooler [21] proposed an adaptation of rational analysis for everyday stimuli, such as emails and news headings. This adaptation can be understood as a statistical mapping between the probability of an encounter and the probability of such an encounter being recalled at a later stage:

$$\text{recall probability} \propto \text{probability of encounter.} \tag{21.6}$$

This means that the more likely a user is to need a created password in the future, the more likely they are to recall it. Vice versa, when a user does not expect to use a password, it is more likely to be forgotten. This simple principle can be used to derive a mathematical model that predicts the password retrieval probability with surprising accuracy [269].

21.3 Information foraging

Information foraging refers to information-seeking activities such as navigating, exploring, comparing, searching, or manipulating information contents in an information space. IFT attempts to explain users' choices in such activities [661]. Example applications include interaction with search engines, web sites, menus, and file systems. The roots of IFT lie in *optimal foraging theory*, originally proposed in biology, which describes the hunting and food search behaviors of animals.

IFT posits that in information-seeking tasks, users exchange their time and effort for information. If a user estimates that there is no interesting information on a web site, the user quits it and does something else to achieve a better payoff. If the user estimates that a particular link opens another web page that may have high information value, it is rational to click it, assuming that there are no better options available. However, if the time cost—the waiting time for the web page to load—associated with that link is high, the user may decide to quit.

IFT proposes that information-seeking behavior develops to maximize the rate of information gained per unit of time or effort invested. Note that the term *information* does not refer to the information-theoretic concept but to subjective interest; here, information means anything that *users* find interesting. This can be quantified in many ways, such as in terms of interest, information need, surprisal, curiosity, learning, or reduction of uncertainty.

A central problem considered by IFT is how users choose between sources of information. To answer this question, the theory introduces the concept of a *patch*. Consider a predatory animal, such as a wolf, with two alternative patches on where to forage or hunt for food. One

patch is a forest while the other patch is a field. Both patches require some travel (i.e., time), although in differing amounts, and both patches offer some calories in return. The forest patch may have deer; the field patch may have rodents. According to optimal foraging theory, an animal changes the patch as soon as the gain in calories decreases to a level that makes it rational to move somewhere else.

Let us assume a wolf has nearly exhausted the food in its forest patch. What would a rational wolf do? Would it stay in the current forest patch, move to a nearby field with rodents offering a low calorie gain, or perhaps travel to another forest patch with deer, which are harder to catch but provide a higher calorie gain?

In *information* foraging, the prey are virtual contents, and the calories are their information. As in optimal foraging, the ecology is assumed to be *patchy*. This means that information is unevenly distributed across regions, or patches, with different information contents. Figure 21.2 shows a simple HCI example with two options.

Similarly, time costs are incurred when moving between patches and when extracting information from a patch. The models of information foraging operate at varying timescales. While individual cognitive actions may take place in the timescale of about 100 ms to 10 s, the adaptation of behaviors may take minutes, hours, or days.

These models have three main components:

Decision: The decision problem that is to be analyzed, such as whether to stay or explore, how long to do something, or how to divide resources among many tasks.

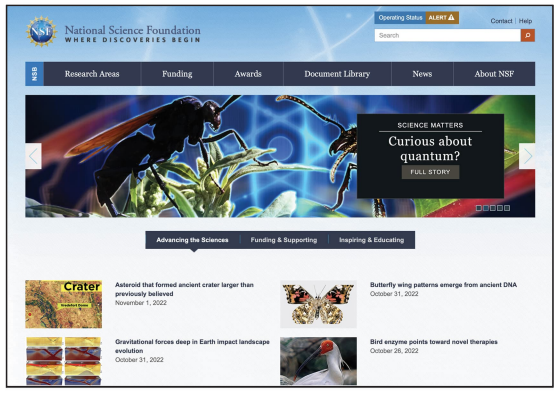
Currency: The dimensions that users compare to assess the choices. Examples of such dimensions include energy, information value, and time. Second, what is the goal of the user, for example, minimization, maximization, or stability?

Constraints: These are the limits of the relationship between decision and currency, such as those posed by the user, interface, or task.

Next, we focus on two well-known information foraging problems: patch departure and diet.

You are writing an essay on what is known about mycobacterial genomes. Should you try browse news articles from a research portal (Patch 1) or a Google search directly from the browser (Patch 2)?

Patch 1



Patch 2

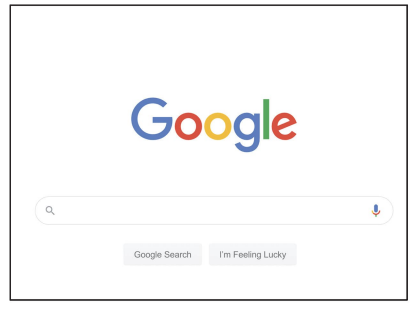


Figure 21.2 Which option should you pick if searching for information on a specialized scientific topic? While a search engine might provide results more quickly, their informational value might be lower compared to searching directly through a research portal.

21.3.1 Patch departure

Whether to enter a patch and how long to spend in it are predicted by the patch departure model. When patches are designed to have different gains in currency, gain rates, and distances between them, the situation becomes complex. The patchiness of digital information means that users must frequently decide on the best way to use their time: Should they stay here and, if not, which of the alternative patches might be the optimal option?

Let us consider a case with P patches. An example would be an interface for selecting movies. The interface shows P movies, and the user is interested in learning about the selection to pick a movie. The decision variable here is the within-patch foraging time. In other words, how long should one consider an option before moving on?

To answer this, we need to estimate the gain of information as a function of how long one stays in a patch. The patch departure model assumes a diminishing rate of information in a patch. The longer it is attended by the user, the less information the user obtains. In other words, the patch gets exhausted over time. Pirolli and Card [661] express this as follows:

$$R = \frac{G}{T_B + T_W} = \frac{G_f - C_f}{T_B + T_W}, \tag{21.7}$$

where G is the net gain, G_f is the gross amount of utility gained, C_f is the total cost of foraging, T_B is the between-patch cost (i.e., the time spent searching or moving to a patch), and T_W is the within-patch cost, that is, the time spent exploiting a patch. A higher R represents improvement.

When should one stop foraging a patch, then? To make the problem realistic, let us assume that each patch has different gain rates and a different between-patch cost. The expression $g_i(t_W)$ denotes the cumulative gain in patch i if t_W time is spent in it.

The special case of linearly increasing patch information is simple to solve. Figure 21.3 shows gain rates for different within-patch times. It illustrates when one should leave a patch.

However, the assumption of linearity is unrealistic: information is almost never extracted linearly. Rather, the relationship is nonlinear. In a well-organized interface, there is a rapid increase in g_i and a quick depletion of information. In a poorly organized interface, the increase in g would be slow at first and fast at the very end. For example, the most important information in a patch is sometimes given at the end, such as in contact forms or poorly designed link descriptions.

Cases where g_i is nonlinear can be solved by using Charnov's marginal value theorem. It states that a forager should stay in a patch as long as the slope of g_i is greater than the average gain rate R for the environment.

This nonlinear case is more challenging. Fortunately, it has a well-known solution when the diminishing returns curve is logarithmic. Figure 21.4 shows an example with two alternative ways to search for an item on a web site. To determine the optimal rates of gain $R*$ for the two cases, graphical reasoning as in Figure 21.4 can be used. A line tangent to the gain function $g_i(t_W)$ is drawn that passes through t_B to the left of the origin. The slope of this tangent is the optimal rate of gain R. More generally, when $g_i(t_W)$ and λ are known, the optimal t_W can be determined analytically. Comparing the two tangents, one concludes that patch 1 is more profitable and should be chosen.

21.3.2 Information diet

Diet models deal with environments that contain a number of potential types of food sources. The organism faces the problem of choosing a diet that optimizes its energy gain per unit cost. Pirolli raised the issue of electronic mail as an example in HCI: An email may come from a variety

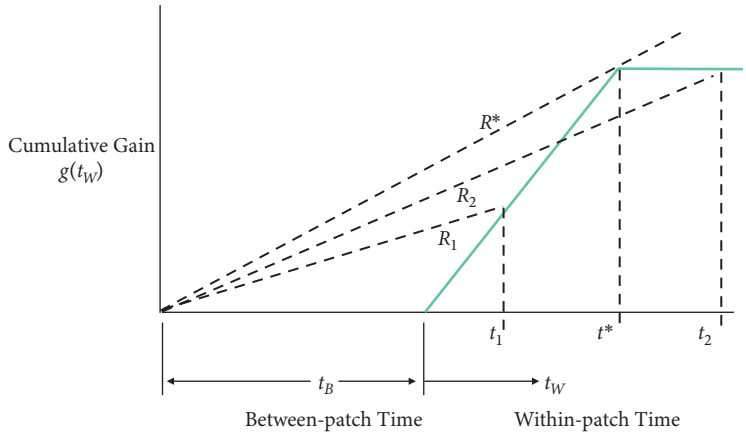


Figure 21.3 A patch with linearly increasing information gain. Here, the optimal time to stay t^* is determined by the moment all information is extracted. The gain rate is then $t^*/(t_b + t^*)$.

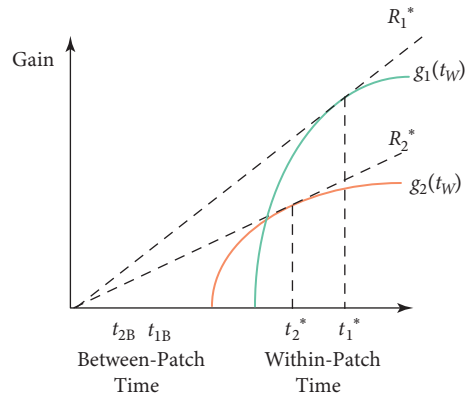


Figure 21.4 Two patches with nonlinearly changing information gains.

of sources that have different arrival rates and profitabilities. Low-profitability junk mail should be ignored if it costs the reader the opportunity to process more profitable mail. The diet of an information forager should also broaden or narrow depending on the prevalence and profitability of information sources.

The idea of the diet model is the following. Let us denote the minimum acceptable gain rate with an information threshold Φ. The diet model states that the forager should choose all patches of the diet that are above this threshold. That is, the patches in the proximal environment are ranked in decreasing order of the information gain rate, and the ones above Φ are picked.

21.3.3 Information scent

Another type of problem the theory considers is how users pick how to get to a *distal goal*. Distal goals are not immediately obtainable; they require intermediate actions. According to the theory,

we need to use *proximal cues* to get to distal goals. For example, imagine you are on the landing page of a web shop. Which link should you click (proximal cue) if you want to get to a page that shows all the mustards on offer (distal goal)?

While optimal foraging theory in biology often assumes *omniscience*, or full observability of the world, Pirolli noted that we often cannot assume this in HCI. The contents of the patches are unknown to the information forager. The informavore must engage in decision-making under uncertainty. From what is locally visible, the informavore must infer what the distant environment may carry. Informavores examine proximal cues to guess what a distal state may carry for them. How do users estimate which patches to choose?

Information scent refers to a user's intuition that a cue in the interface represents the information needed (see Figure 21.5. for an example). It is an estimation of relevance based on a proximal cue. Consider, for example, the task of finding the address of a company on an unfamiliar web site. There are many ways this could be solved. One might directly start by typing "contact information" into a search field. If the first results are fruitless, the information scent is low, and one would be wise to switch to a different strategy, for example, a visual search of links. Many times when searching for links, there is no direct match, although there may be a few promising candidates. In our scenario of finding contact information, any of the following would have some "scent" but no perfect match: "Press," "Contact," or "About us."

IFT has several implications for user interface design:

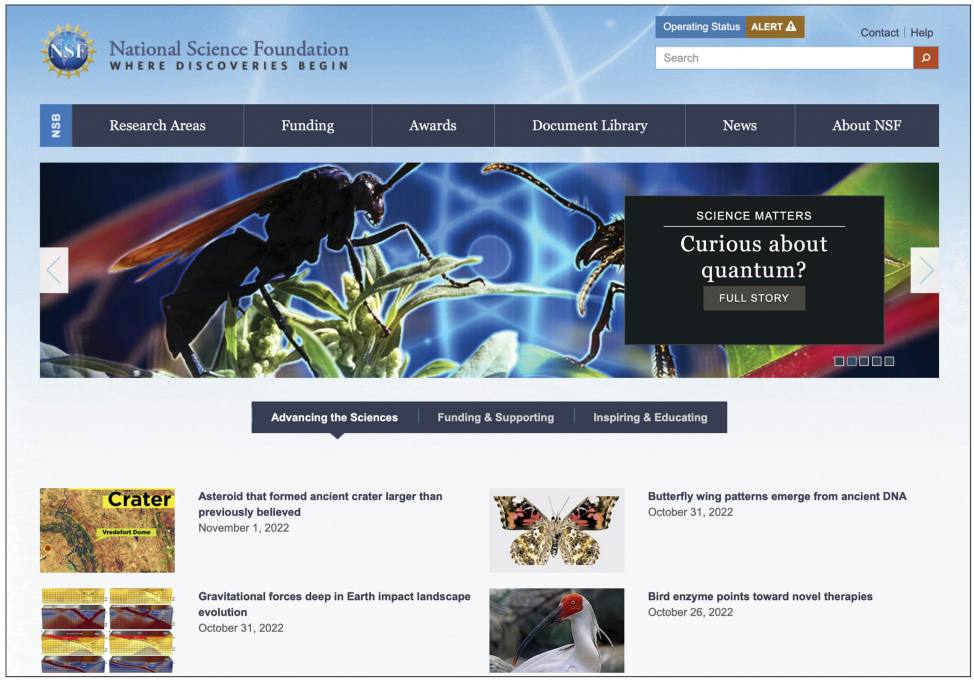
1. Decreasing the between-patch transition times T_B will increase the gain rate. This can be done by decreasing the system latency or the time needed to complete the steps.
2. Decreasing between-patch times will also make users more willing to use the different options of the interface, as more patches will be above the threshold Φ.
3. Increasing gains in a patch g_i will increase the probability of users choosing the patch and spending longer in it.
4. Users can be deliberately directed to use a particular means by increasing its t_B. This "inverse usability" is used by commercial sites that want to promote certain options.
5. Gain functions that peak and asymptote quickly better support information foraging, as users can move to better patches more quickly. In other words, most information should be made quickly absorbable, for example, by using overviews or figures.

21.4 Computational rationality

In HCI, rewards are typically *sparse*; that is, several actions need to be taken to get to a rewarding state. For example, clicking a link may not take you directly to the page you want, yet the click is rational if, via that page, you get closer to your target page. Furthermore, some rewards are *very* sparse. Consider, for example, starting to use a social media app with the goal of becoming an influencer. This requires a long-term commitment to something that can go wrong in many ways and may not even pay off. In other words, in HCI, gratifications are typically delayed.

To deal with the problem of sparse rewards, we need to expand our account of rationality to cover sequential decision-making. The user starts in some state s_1 and picks an action, say a_5, which then takes the user to, say, state s_8. Here, the user picks a_9, transitioning to state s_{39}. A rational user would have chosen utility-maximizing actions. However, how would a user know which actions maximize utility?

(a)



(b)

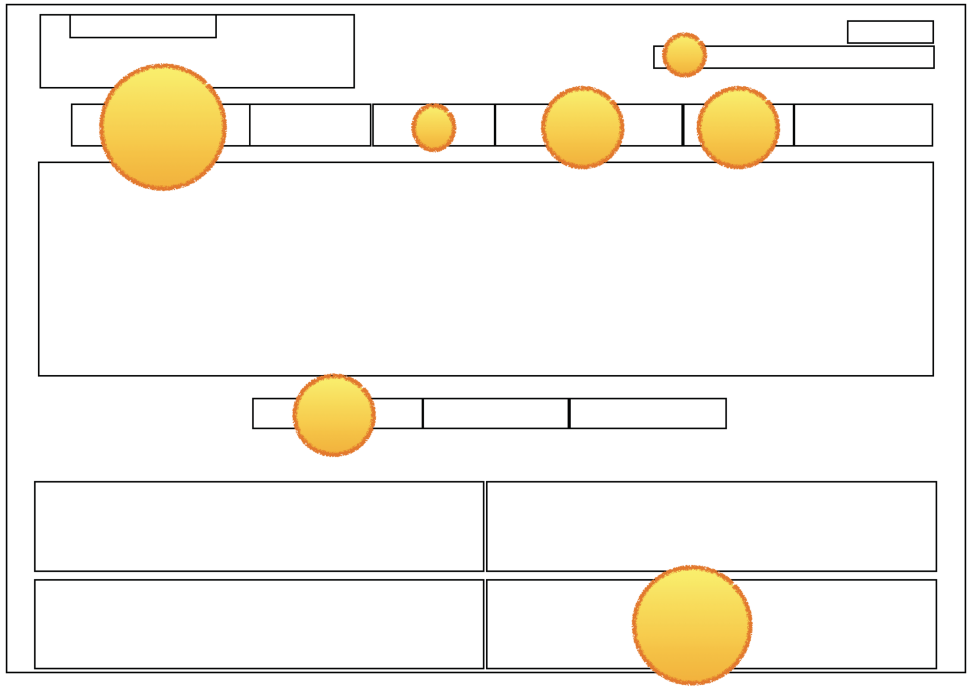


Figure 21.5 If you are looking for, say, articles on mycobacteria, certain patches have a higher information scent than others.

Computational rationality is a theory and a modeling approach rooted in bounded rationality and bounded optimality. Recent applications include typing (Figure 21.7), pointing, driving, multitasking, menu selection, and visual search. Its core assumption is that users act in accordance with what they *believe* is best for them. These beliefs are bounded by the limits of their cognition; for example, forgetting happens. They are also limited by their experience of the task environment [475]. Users do not know the true state of the world; cognition forms beliefs about it based on experience. As we learned in Chapter 5, there are many belief-forming capabilities in human cognition: perception, multiple memory systems, and the ability to reason and infer.

The theory assumes that users are "computationally rational": When picking an action—or deciding how to get from the present state to a state with positive rewards—users are as rational as their cognition allows. Users act based on their often inaccurate and partial beliefs, which they have formed via experience. Hence, unlike in IFT, in computational rationality *both* environment- and cognition-determined bounds are present. For designers, this means that design does not *determine* user behavior. It changes the external environment of the user, which the user forms beliefs about via experience.

Paper Example 21.4.1: Menu search as computationally rational behavior

To illustrate computational rationality, let us consider the task of searching for an item in a menu (Figure 21.6). For example, consider a user looking for the item "Print" in an application menu. Users often make choices on where to look and what to click quickly and without conscious deliberation. Computational rationality can explain such choices and predict the gaze and cursor patterns of users [147].

The decision-making problem the user faces is the following: At any given time, the user can perceive accurately only a part of the menu—whatever can be gathered in a fixation. Even if we think we see a menu "fully," we need to move our eyes to find the item we are looking for. After the first fixation, we have several options on where to fixate next to find the item. However, every fixation wastes time (negative reward), so the user wants to minimize the number of unnecessary fixations. When the user finds the target, a positive reward is allocated. A sequence of fixations must be chosen to land on the target, ideally wasting as little time as possible.

How does cognition solve this challenge? It can be approached in a few ways, for example, by following a simple heuristic strategy: "Start at the top and read each item one at a time." However, this strategy is inefficient: If the menu is large, a lot of time would be wasted looking at irrelevant items. Other strategies include (1) guessing where the target could be, (2) using a visual cue (e.g., a separator line) to jump to a promising area (e.g., the end of the menu), or (3) trying to recall the target's location and directly look at that location.

The optimal strategy depends on the design of the menu and how much experience the user has: If the menu is arranged randomly, guessing may be as good a strategy as any. However, if there is any structure to exploit, the theory assumes the users will pick a strategy to exploit that structure. For example, if the menu is alphabetically organized, users can jump (fixate) closer to the target and start looking at options serially there. If the menu is semantically grouped, they can try to first identify the group where the item is and then focus the search on that group. If the user is experienced with the menu, the position of the item may be recalled from long-term memory.

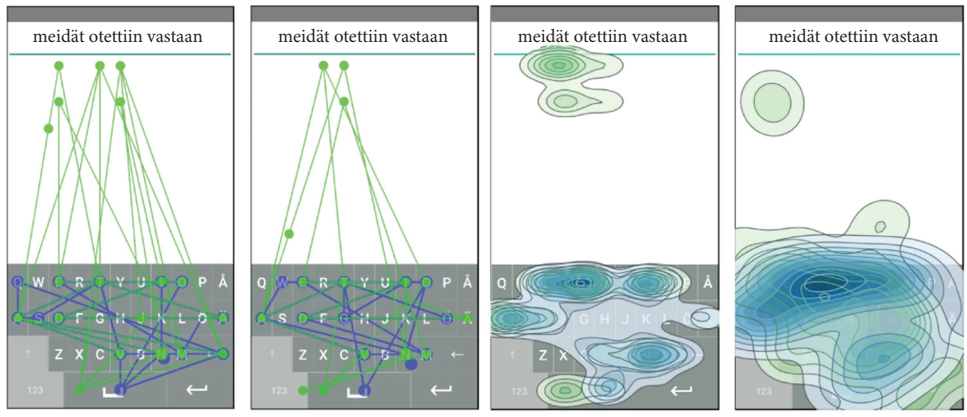
Still, how do users decide which strategy to follow? They make estimates on which strategy offers the best payoff. They estimate how quickly they might find the target, given what they know about it. Such decisions happen unconsciously; users "just do it." However, with eye trackers, we can expose such strategies.

Figure 21.6 The Markov decision process is a formal description of a sequential decision-making process. An agent must sequentially decide which action to choose to maximize cumulative reward. For example, even if we think we see a menu "fully" (left), we need to move our eyes—often several times—to find the item we are looking for. After the first fixation (middle), we have several options (actions, a_x) on where to fixate next to find the item. However, every fixation wastes time (negative reward), so the agent wants to minimize the number of unnecessary fixations. When the agent finds the target, a positive reward is allocated.

21.4.1 Modeling interactions

Computational rationality allows for building computational models that generate predictions of user behavior. First, we formally define the sequential decision-making that the user is facing in interaction. The possible states are enumerated, for example, the beliefs or perceptions about the user interface. The possible actions in each state are then enumerated. They are typically commands to the motor system, such as a command to move the eyes or the hand. Then, the user's rewards—both positive and negative—are modeled.

A solution strategy—or "policy" in reinforcement learning (RL) terms—can now be approximated by using RL, a machine learning method for learning from experience. Because of the complexity of the decision-making problem, there is no analytical solution; the solution must be approximated through trial and error. In practice, this happens by allowing the RL agent to trial numerous (typically millions) possible behaviors in a simulated environment. Eventually, the RL agent learns a *policy* for which action maximizes the expected value in each state. Here, a policy is a generalization of the concept of choice we have discussed previously. In one-shot choice, we have a set of options and we can choose once; by contrast, a policy describes how a user would choose across all possible situations in a task. Technically, it is expressed as a probability function that describes how users choose *actions* in the different *states* of an environment.



(a) Model typing, with some errors and no automatic error -correction

(b) Model typing, with no errors and with automatic error-correction

(c) Model typing, no errors and no automatic error-correction

(d) Human typing, no errors and no automatic error-correction

Figure 21.7 When typing, a user needs to decide when to look at the keyboard and when to look at the text display to spot possible typos. Visual attention is needed to guide the fingers to key locations. When it is not allocated to the fingers, uncertainty increases about their position. However, highly learned key locations can be pushed without guiding the fingers all the time. Computational rationality provides a way to model this decision-making problem and generate patterns of eye and hand movements that have a close correspondence with human data [392].

For those readers interested in machine learning, it is noteworthy that in order to enable the use of RL, the decision-making problem is first formulated as a sequential decision-making problem called the Markov decision process (MDP). MDP is a formalism that originates from studies of sequential decision-making in artificial intelligence and operations research. Instead of the choice between n actions, MDP deals with environments where rewards are delayed (or distal). This requires an ability to plan actions as part of *sequences* instead of one-shot choices. This is necessary for many applications in HCI where the environment is complex. For example, consider the task of writing an essay for homework. The rewards are mostly stacked at the end of the task, when the essay is submitted for review. Similarly, searching for a menu item has mostly a negative reward (the time cost) until the item is found and clicked (Figure 21.6).

Formally, MPD is defined as the tuple $(S, A, P(a, s, s'), R(a, s, s'), \gamma)$, where:

- S is a finite set of states and s is the current state;
- A is a finite set of actions;
- $P(a, s, s')$ defines the transition probabilities between states s and s';
- R defines the rewards obtained in each state.

The state can be changed to $s' \in S$ by an action $a \in A$. After each action, the agent receives a reward $r = R(a, s, s')$. States in MDP can also be defined in terms of *beliefs* instead of external states, forming a belief MDP. The beliefs may be about technology or the person's internal state. For example, they may include a memory of having visited a state earlier. The policy $\pi(s)$ defines which action is performed in each state.

For example, consider a linear menu with n commands. Formally, the MDP contains the following elements:

- S are fixated-upon commands in the menu;
- A are decisions that command what to fixate on next;
- $P(a, s, s')$ defines the next fixation point given the action;
- R yields a positive reward for finding the target (e.g., 100) and a negative reward for every wrong command fixated (e.g., 1).

The MDP can be solved via RL. The result is a prediction of the user's strategy for looking through the menu. The MDP predicts how the user's behavior changes depending on the given conditions. For example, when the menu is arranged randomly, systematic scanning emerges when the agent is trained with random target locations. When the menu is organized into semantic groups, the optimal policy is to inspect the first items of a group and jump to the next group if the target is not in the first group.

21.5 Are users rational?

Are people rational and maximizing utility? The answer depends on the viewpoint. Most of the time, people fail to reach the optimal choice. We procrastinate and choose shortcuts that damage our interests later on. In this sense, we are not rational. A competing explanation is that we are not rational but our behavior is driven by situations. At the opportune moment, we change our plans and goals [804]. However, rationality is not an inherent contradiction of the view of behavior as situationally shaped. In fact, that our behavior is rational is demonstrated by our ability to *adapt* to new, unforeseen circumstances.

Consider browsing a news feed polluted by numerous ads. Because of the low intrinsic value of those ads, it is rational to not attend to them ("ad blindness"). With time, we become even better at skipping ads. However, let us say we bump into one advertisement that *is* interesting. This may change our course of action from that point onward because the expected value of those ads is now higher. If our behavior were not adaptive, we would be at the mercy of convenient options in our environment.

Theories of rationality have increased our understanding of how users fail to be optimal. They fail because their beliefs are wrong. They fail because of their limited cognitive and other capabilities. They fail because they often do not have the cognitive resources or time to seek optimal solutions. We learned in Chapter 5 about multiple cognitive limits, for example, related to attention, working memory, and long-term memory. Under constrained conditions, even if users try to act rationally, they end up with choices that may seem suboptimal or lazy. However, they are still "resource-rational": We do our best, given our cognitive resources [480]. For example, while a user may *want* to find the best privacy settings for an application, time pressure, cognitive effort, or perceived incompetence (e.g., inability to comprehend the options) may limit performance.

Users also fail because the rewards are too distant in the future. If you have never completed a major project, it is easy to procrastinate. For example, the immediate reward from skimming social media feeds is higher than the reward from starting something that will take hours to complete. Theories of rationality offer us insight into this. The further in the future the state that

contains the reward, the longer the chain of actions that are needed, and therefore the harder it is to identify a reasonable policy. Computational rationality explains these seemingly irrational behaviors.

To sum up, descriptive theories of rationality do not contradict the idea that users behave in a way that appears lazy or careless to an observer. For example, a user picking weak passwords simply prioritizes other goals, such as minimizing effort. What appears to be lazy behavior can be rational adaptation to achieve *other* goals.

Theories of rationality can be used to inform the design of information environments, addressing considerations such as how to distribute and shape information. The application of such theories requires careful analysis of the reward–cost structure of the interactive environment. When this structure is known, the environment can be changed to the benefit of the user in multiple ways:

1. Finding sweet spots of design parametrically, like we did for the search query length.
2. Computationally searching for the best way to structure the environment.
3. Studying how robust the environment (design) is to changes in the user's interests (rewards) or capabilities.

An open challenge for these models is how to model users' *intrinsic* motivations, such as motivations for learning or enjoyment, which are hard to observe.

Summary

- Theories of rationality do not describe what a user has done but ask what that user *could* have done. Rational behavior refers to behavior that seeks to maximize the expected utility to the user.
- Satisficing refers to boundedly rational behavior: A user picks the first satisfactory (good enough) option instead of trying to pick the best option by exhaustively assessing all options.
- Rational behavior refers to attempts to take the best course of action within the given constraints. These bounds include those of the external environment (e.g., the user interface and its structure) and those of the internal environment (e.g., cognitive limits).
- Theories of rationality can make quantitative predictions on user behavior in settings where the user's environment and goals (rewards) are well known.

Exercises

1. Optimal querying. Solve the optimal querying problem given in Paper Example 21.1.1. Plot the ratio of profit to cost as a function of query length (x-axis) and solve for W^*.
2. Foraging theory. Consider a card interface in an in-flight entertainment system. Each view consists of a 3 × 8 matrix of cards, each showing a movie. Let us assume that all movies contain the same amount of information and use foraging theory to estimate how long a user should skim one item. Given $a = 0.2$ and $b = 0.3$, compute t_w^* using the patch departure model.
3. Markov decision process. Define an MDP tuple for menu interaction. Assume a simple linear menu with 10 commands. The target command is in a random position. The agent starts from

the top command, and their task is to find the target. There are 11 possible actions: fixate on any of the items (1, . . ., 10) or click. Moreover, there are 13 possible states, one for the item being fixated (1, . . ., 10) and one for its status (unknown, not-target, target). A hefty reward is awarded for clicking the target, and a small negative penalty is administered for wasting time, that is, for every action that is not clicking the target. When this MDP is solved, what kind of policy emerges?

4. Rationality. What are some behaviors in HCI that can be argued to be (a) rational, (b) boundedly rational, and (c) irrational (i.e., not explained by theories of rationality)?