# Progress Indicators in Web Surveys Reconsidered — A General Progress Algorithm

Thomas M. Prinz, Raphael Bernhardt, Jan Plötner, and Anja Vetterlein
Course Evaluation Service, Friedrich Schiller University Jena
Jena, Germany
e-mail: {Thomas.Prinz, Raphael.Bernhardt, Jan.Ploetner, Anja.Vetterlein}@uni-jena.de

***Abstract*—The inclusion of a progress indicator seems to be a simple part during the construction of a web survey. This paper shows that this is only true for linear surveys and does not hold true for surveys with adaptivity (branches). Therefore, we introduce a general model, which fits almost any kind of survey. Based on this model, the difficulties of progress computation are shown. As a solution, this paper proposes a general equation and algorithm to compute the progress dynamically. This algorithm predicts the number of remaining items for each page. Since the remaining items depend on the actual path through the survey, the algorithm was generalized by a so-called selection operator allowing different prediction strategies. With the help of this algorithm, it is possible to compare different strategies for a given survey and to select the best one. In examples, two prediction strategies are introduced which estimate either the most or the least remaining items. The paper concludes with a comparison of both strategies showing that the choice of the best prediction strategy is not trivial and depends highly on the structure of the survey.**

***Keywords–Progress Indicator; Computation; Adaptive Surveys; Web Surveys; Questionnaire Model.***

## I. INTRODUCTION

Most web surveys use Progress Indicators (PIs) to give the participants feedback about the degree of completion. Such a feedback should motivate the participants to complete the survey and, therefore, consequently reduce the break-off rates [1][p. 146]. Usually, a PI shows the progress of a task between its start and its completion. It is common practice to calculate the progress in percentage and use $0\%$ at its start and $100\%$ at its end. The *current* progress is a number between both values.

A PI seems to be a simple feature during the construction of a web survey in which not much time and resources should be invested. For *linear* surveys this is obviously true — a linear survey has no adaptivity, all questions have a specific order, and this order is the same for each participant. But the computation of the progress is difficult for surveys with *adaptivity* (branches). In such surveys, a simple computation approach produces *jumps* [1] in the progress since some questions are not shown to each participant. Sometimes those jumps are large and could be demotivating and perceived as untrustworthy by the participant. But the progress feedback should be truthful to the subjective sense [2, p. 757]. For this reason, we need a trustworthy calculation for those complex surveys.

Since there are a lot of survey tools currently on the marked, comparing their PI approaches seemed to be a good basis for a general progress calculation in complex surveys. However, either those tools explicitly do not support PIs in complex surveys like *Survey Monkey* [3], or their approaches remain guarded: the consideration of other survey tools does not help.

The Ph.D. thesis of Kaczmirek [1] is the best starting point for the development of such an algorithm to the authors' knowledge. In his thesis, he developed the attempt of a PI computation approach for complex surveys, which unfortunately has not garnered a lot of attention in research. One reason could be the neglect to describe his approach in a concrete algorithm with a formal model.

We have developed a formal model for questionnaires used in our own survey tool Coast [4]. The model is based on mathematical graphs, which help to understand the structure of a survey and create the possibility to apply well-known algorithms of graph theory as shown in previous work [5]. Based on this model, this paper defines a general algorithm for the computation of the progress in surveys. This algorithm allows arbitrary strategies to handle complex questionnaires and compare them. As an example, this paper formalizes the two proposed strategies of Kaczmirek (used in its Ph.D. thesis) — using the maximal and minimal number of remaining items. A comparison of both strategies shows that choosing the best strategy for a given questionnaire is not trivial. As a result, the proposed strategy of Kaczmirek — taking the minimum of remaining items — should be taken with caution. There is the need for more research for defining more prediction strategies and for choosing a best fitting strategy for each survey.

This paper considers current PI research at first (Section II). In Section III, our questionnaire model is introduced, which is based on mathematical graphs. Afterwards, the general algorithm for computing the progress is derived in Section IV. In Section V, this algorithm is used to explain the two PI computation approaches of Kaczmirek. Furthermore, both approaches are compared and the difficulties of selecting the right approach are shown. The paper closes with a conclusion and a look into future work (Section VI).

## II. STATE OF THE ART

Human-Computer Interaction (HCI) has considered PIs for a long time. It focuses on the duration of a task and how PIs help the user to be aware of that duration. A result is that people prefer to have a PI in comparison to not having one and that the perceived duration depends on the progress speed [6][7]. Harrison et al. found out that a user suggests a task runs faster if the PI is faster at the end of a task [8].

 ISBN: 978-1-61208-686-6 

However, PIs in web surveys are different from those classic PIs in HCI. Villar et al. state that the tasks in web surveys are usually longer, and that the user can influence the PI and has to focus on the task. Furthermore, users of classic computer tasks want the goal of the progress, e. g., transferred money, loaded files, etc.; whereas web survey participants do not necessarily have an interest on the result. Because of those differences, PIs in web surveys should be considered with another focus as PIs in classic HCI [2].

In 1998, Dillman et al. defined some principles for the design of a web survey. These principles include PIs with less implementation costs [9]. Some years later, Conrad et al. considered different speeds of PIs for the first time: 1) *constant*, 2) *fast-to-slow*, and 3) *slow-to-fast* speeds [10]. The speed of a PI varies if there is a difference between the *displayed* and the *true* progress. The *true* progress is the progress when the remaining number of questions would be known at each point. If the difference of the displayed to the true progress is always positive, the displayed progress runs faster at the beginning and has to become slower at the end (fast-to-slow). If the difference is always negative, the displayed progress is slower at the start and becomes faster at the end (slow-to-fast).

The work of Conrad et al. was the first consideration of a divergence between the displayed and the true progress. It is especially interesting since they take notice of a correlation between the break-off rates and the speed of the PIs. It seems to be that a slow-to-fast progress is more discouraging and causes a higher break-off rate. On the contrary, a fast-to-slow progress seems to encourage the participants to finish the survey. These results were supported by Matzat et al. [11] and succeeding studies of Conrad et al. [12]. Villar et al. combined all those studies in a meta-analysis. Their results were: 1) For most studies, the decrease of break-offs is significant when showing a fast-to-slow PI instead of having none. 2) The break-off rates increases significantly for most studies if a slow-to-fast PI is applied. 3) It poses an ethical problem if the speed of the PI is manipulated on purpose in order to deceive the participants. Sometimes, however, the varying speed is not deceptive if it tries to accurately mirror the true progress [2]. A study about 25,000 real world surveys seems to state the contrary, i. e., that the PI has no effect on the break-offs [13].

Although there are a lot of studies considering the differences in PI speeds, there is missing research, which consider, *how* to compute the progress accurately. This question becomes important especially for surveys with high adaptivity. To the best knowledge of the authors, the thesis of Kaczmirek [1] is the only published work, which tries to answer this question. In one of his studies, he explains that a simple computation approach produces jumps and, therefore, he introduced a new *dynamic* strategy, which converges more to the true progress. A little study in his thesis suggests that the application of his dynamic approach does lead to less break-offs as the simple approach.

A problem of all progress computation approaches in adaptive web surveys is the logical lack of knowledge, which "*path*" through the survey a participant takes. Therefore, the true progress is only known *after* the participation. Kaczmirek [1] proposes two strategies to predict the remaining number of pages (and therefore the path): either the 1) maximum or the 2) minimum number of remaining pages is taken for the calculation. In a little study in his thesis, the break-off rates of the minimum are less than those of the maximum approach. This corresponds to the results of Villar et al. since the minimum strategy is similar to a fast-to-slow and the maximum approach is similar to a slow-to-fast PI. However, instead of manipulating the progress speed by design, Kaczmirek is not deceptive as he does it to approximate the true progress as good as possible.

## III. Questionnaire Model

It is important to know how surveys are structured for the definition of an algorithm for the computation of the progress. Since surveys are like computer programs, the structure can be described precisely as a *directed graph*. A *directed graph* (or *digraph*) $G = (N, E)$ consists of a set $N = N(G)$ of *nodes* and of a set $E = E(G)$ of *edges*. $E$ forms a set of ordered pairs, $E \subseteq N \times N$ [14, pp. 432]. The sets $\triangleright n$ and $n \triangleleft$ describe the sets of all incoming and outgoing edges of a node $n$, respectively, i. e., $\triangleright n = \{(n', n) \in E\}$ and $n \triangleleft = \{(n, n') \in E\}$.

Since nodes are connected by edges, it is possible to travel from one node to another node via a sequence of edges. Such a sequence of edges is called a *path*. In a digraph $G = (N, E)$, a sequence $W = (n_0, \ldots, n_m)$, $m \geq 0$, of nodes, $n_0, \ldots, n_m \in N$, is a *path*, if each node of the sequence is connected via an edge to the next node in the sequence: $\forall\ 0 \leq i < m \colon (n_i, n_{i+1}) \in E$ [15, p. 1180]. A path is called *acyclic* if all nodes on the path are pairwise different. A digraph is *acyclic* if each of its paths is acyclic. Otherwise, the digraph contains a loop and is cyclic. A digraph is *connected* if its undirected pendant (adding an edge $(a, b)$ for each edge $(b, a)$) has a path between each two (different) nodes [16, S. 547].

In our model, a survey is an elicitation that uses a *questionnaire* as measurement method. Such a questionnaire consists of *items* [17, p. 18]. An item is a concrete question sometimes with some answer possibilities. In almost all questionnaires, more than one item is presented in sequence. Mostly, they are grouped thematically on *pages*.

*Definition 3.1 (Page):* A *page* $S = \{i_1, i_2, \ldots\}$ ($S$ for *s*heet of paper) is a set of items $i_1, i_2, \ldots$.

It is known from test theory that the structure of a questionnaire influences the measurement results [18, S. 68 ff.]. For this reason, a questionnaire cannot be defined only using a set of pages since this set is unsorted. We have to define an order of the pages. From our experience, it is promising to describe the structure of a questionnaire as an acyclic, connected digraph:

*Definition 3.2 (Q-Graph):* A *Q-graph* $Q$ is an acyclic, connected digraph $(\mathbb{S}, \mathbb{E})$ with a set of pages $\mathbb{S}$ and a set of *edges* $\mathbb{E}$ connecting the pages. A Q-graph $Q$ has exactly one page without any incoming edges, the *start page*, and exact one page without any outgoing edge, the *end page*.

The previous definition describes the *structure* of a questionnaire as a Q-graph. From a detailed described Q-graph, a

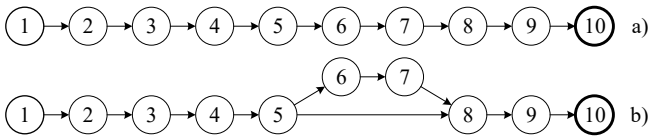


Figure 1. A linear *a)* and non-linear *b)* Q-graph

web survey can be automatically derived [4]. However, this derivation is not part of this work due to the lack of space.

## IV. A General Progress Algorithm

In the last section, a Q-graph described the structure of a questionnaire as a graph. The complexity of the computation seems to depend on the complexity of the describing Q-graph. The Q-graph can be either simple or complex.

### A. Simple Case

Sometimes, a questionnaire is a simple sequence of pages without any adaptivity and filters. Its Q-graph is, therefore, a chain of pages as illustrated in Figure 1 *a)*. More formally, each page (except the end page) has exactly one outgoing edge. Such a Q-graph is called *linear* in the following.

In the case of a linear Q-graph $Q$, it is possible to assign a position to each page depending on its distance back to the start page: the start page gets position 1, the first page after the start page gets position 2, etc. Finally, the end page has position $|\mathbb{S}(Q)|$. We write $S_t$ for the page on position $t$.

If a participant is on a page $S_t$ of a Q-graph, then there are two interesting moments regarding the progress: 1) the moment the participant *reaches* the page and 2) the moment the participant *leaves* the page. Obviously, the progress when reaching the *current* page is equal to the progress when leaving the *previous* page. It is assumed in the following, that the progress $\rho(S_t)$ on page $S_t$ reflects the progress at the moment, the participant *reaches* the page $S_t$. Since, there is no previous page of the start page, the progress $\rho(S_0)$ is defined as 0 at position $t = 0$.

For example, if a participant reaches page $S_7$ of the Q-graph in Figure 1 *a)*, already $^7/_{10}$ of the Q-graph are processed and the progress $\rho(S_7)$ is 70%. After leaving this page, the progress is $^8/_{10} = 80\% = \rho(S_8)$. If a participant just started a questionnaire, then the progress is $\rho(S_0) = 0\%$.

Since there is a linear association between the page position and the progress, the progress $\rho(S_t)$ on a page $S_t$ of a Q-graph $Q$ can be computed with:

$$\rho(S_t) = \frac{t}{|\mathbb{S}(Q)|} \tag{1}$$

This equation was proposed by Kaczmirek [1][p. 147] and he called it "*static*" calculation approach. However, it has its limitation if the progress should be computed in more precision by the number of processed items. After finishing a page $S_t$, $|S_t|$ items were processed ($|S_t| = |\{item_0, \ldots, item_m\}| = m$, $m \geq 0$, since a page is a set of items regarding Definition 3.1). Finishing page $S_t$ expands the progress by a term $\frac{|S_t|}{|\bigcup_{S \in \mathbb{S}(Q)} S|}$, where $|\bigcup_{S \in \mathbb{S}(Q)} S|$ is the total number of all items of the Q-graph. This term is added to the progress of the previous page ($\rho(S_{t-1})$). It results in a general, recursive and non-recursive equation to compute the progress of a linear Q-graph with item precision:

$$\rho(S_t) = \rho(S_{t-1}) + \frac{|S_t|}{|\bigcup_{S \in \mathbb{S}(Q)} S|} = \frac{\sum_{i=1}^{t} |S_i|}{|\bigcup_{S \in \mathbb{S}(Q)} S|} \tag{2}$$

This equation cannot be simplified as long as the number of items varies from page to page. If the progress should be computed only with page precision, it can be assumed that on each page lies only one item. The formula simplifies to:

$$\rho(S_t) = \rho(S_{t-1}) + \frac{1}{|\mathbb{S}(Q)|} = \sum_{1}^{t} \frac{1}{|\mathbb{S}(Q)|} = \frac{t}{|\mathbb{S}(Q)|}$$

which is equal to the equation of Kaczmirek [1][p. 147].

### B. Complex Case

Now, it is assumed, the Q-graph is non-linear (adaptive), i. e., there is at least one page in the graph, which has at least two outgoing edges (cf. Figure 1 *b)*). As a consequence, the paths throughout the Q-graph depend on the answers given by the participants resulting in multiple possible paths. Each possible path can have a different number of pages and items. Therefore, it is *not* possible any more to use the total number of pages $|\mathbb{S}(Q)|$ or the total number of items $|\bigcup_{S \in \mathbb{S}(Q)} S|$ for the computation of the progress.

Take the Q-graph of Figure 1 *b)* as an example. The total number of pages is 10. For the moment, it is assumed that each page consists of a single item and that we apply (2) for linear Q-graphs. If a participant answers the questions on the first five pages, the progress increases to $^5/_{10} = 50\%$. If the participant takes the lower path, the path shortens to 8 pages. In this case, the participant reaches the end page with an incorrect progress of 80%. Another variant is to skip the progress of the pages 6 and 7 such that the progress finishes with 100%. But this produces a big jump in the progress, which can be misleading.

Kaczmirek proposes a *dynamic* computation of the progress in questionnaires with adaptivity. His approach considers the remaining progress and the contribution of the current page to the overall progress [1, p. 148]. His resulting equation resembles (2):

$$\rho(S_t) = \rho(S_{t-1}) + \frac{1 - \rho(S_{t-1})}{|\text{remaining pages}|}$$

The equation flattens the jumps and, furthermore, allows a dynamic changing of the number of remaining pages. Therefore, this solution allows to compute the progress for non-linear Q-graphs. However, we identified some disadvantages on the above equation:

1) The equation considers only page precision rather than item precision (except there is only one item on each page).
2) The term *remaining pages* is vague and has to be discussed in more detail.

 ISBN: 978-1-61208-686-6 

To overcome the first disadvantage, we reformulate the equation as follows:

$$\rho(S_t) = \rho(S_{t-1}) + |S_t| \frac{1 - \rho(S_{t-1})}{|\text{remaining items}|} \quad (3)$$

This equation multiplies the number of items on the current page, $|S_t|$, and considers the remaining *items* instead of pages. The number of remaining items includes the items of the current page, $|S_t|$, as we still consider the progress at the moment when reaching a page. In other words, the term $\frac{1-\rho(S_{t-1})}{|\text{remaining items}|}$ describes the influence for a page with a single item, whereas $|S_t|\frac{1-\rho(S_{t-1})}{|\text{remaining items}|}$ describes the influence of all items on the current page on the progress. Although the progress actually depends on the current answers of the participant, the equation would become long and difficult to read. Therefore, we ignore an explicit inclusion of the current answers into the equations.

In (3), a detailed description of the *remaining items* is still missing. The difficulty of describing the number of remaining items is that it is highly dependent on the answers, a participant has already given, and in the case of future branching paths on the answers, the participant has to give in future. As it is impossible to forecast which path of the Q-graph will be taken by the participant, it is impossible to predict the remaining items accurately.

The exact number of remaining items on the current page $S_t$ depends on the remaining, individual path $W$ to the end page. Since there may be different paths $W_1, W_2, \ldots, W_m$, $m \geq 1$, to the end page with the *same* number of remaining items, it is *not* of interest to predict the exact path to the end page rather than that number of remaining items. This remaining items prediction function $rem$ depends on the current page $S_t$. The resulting equation for computing the progress in non-linear Q-graphs is:

$$\rho(S_t) = \rho(S_{t-1}) + |S_t| \frac{1 - \rho(S_{t-1})}{rem(S_t)} \quad (4)$$

Theorem 1 in the appendix of this paper shows that (4) is equal to (2) for linear Q-graphs.

### C. Compute Remaining Items

While most of the last equation can be computed easily during the survey of a participant, the remaining items function $rem$ is still challenging. In the following, let $Q$ be a Q-graph with an end page $E$ and $S_t$ the current page.

Basically, there are three different situations one can encounter on each page $S_t$: either $S_t$ has *a)* no successor page, it has *b)* exactly one direct successor, or it has *c)* at least two direct successors. This is illustrated in Figure 2.

In the first situation *a)*, the participant reached the end of the Q-graph. Therefore, the number of remaining items is known as it is equal to the number of items on the end page ($n = |E|$), i. e., $rem(E) = |E|$ (cf. Figure 2 *a)*).

In the second situation *b)*, the current page $S_t$ has exactly one direct successor. If the remaining items are known for the direct successor as $a$, then it is also known for the current page as $a + n$, where $n = |S_t|$ (cf. Figure 2 *b)*).

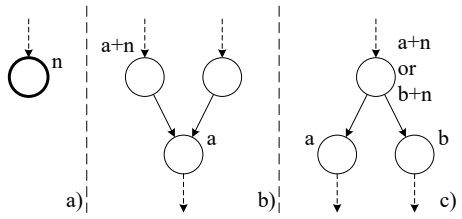


Figure 2. Three situations: *a)* no successor, *b)* one direct successor, *c)* two direct successors

In the last case *c)*, the difficulties arise since the current page $S_t$ has at least two direct successors. Assume the remaining items on one successor is $a$ and on the other successor is $b$ (in the simple case of two direct successors). The current page now has either $a + n$ or $b + n$ remaining items, $n = |S_t|$ (cf. Figure 2 *c)*). The exact remaining items are only known if $a = b$. Otherwise, we have to predict, which direct successor page of $S_t$ will be visited by the participant and, therefore, which remaining items $a + n$ or $b + n$ are taken for the computation of the progress.

For such a prediction, there are different *strategies*. For example, Kaczmirek [1] proposes two strategies: always take 1) the maximum or 2) the minimum number of remaining items. Both strategies are considered in the next section as examples. At this stage, a selection operator $\sqcup$ is introduced, which combines the solutions $a+n$ and $b+n$, e. g., $\sqcup$ can be the minimum, maximum, or another arbitrary function. The introduction of this selection operator $\sqcup$ makes it possible to define a general algorithm:

The prediction algorithm gets a Q-graph $Q$ and a selection operator $\sqcup$ as input. We emphasize that $\sqcup$ is an *input* parameter of the algorithm and can be chosen individually.

At first, each page gets an initial value 0 as remaining items (the steps of the algorithm can be followed at Figure 3). Afterwards, the pages of the Q-graph will be put into a $work\ list$ in an arbitrary order. This work list contains all pages *without* computed remaining items. Furthermore, there is a set $visited$, which contains all *computed* pages. In a while-loop, the algorithm extracts the first page $S$ of the work list and tries to compute the remaining items. This is only possible for $S$ if all of its direct successors are in $visited$. The computation then follows our previous ideas corresponding to Figure 2. Otherwise, if $S$ cannot be computed, $S$ will be placed on the end of the work list. Finally, the algorithm terminates when the work list is empty, i. e., when each page was computed.

With Figure 3, we have found a general algorithm to compute the remaining items for each page of a Q-graph and for arbitrary strategies. Combined with (4), the progress can be calculated for arbitrary Q-graphs and selection operators allowing the comparisons of different strategies. This is explained in detail on two examples in the next section. Since the Q-graph is acyclic, the algorithm of Figure 3 always terminates. It can be easily checked, that the asymptotic runtime complexity of the algorithm depends on the $\sqcup$-operator if the pages are put topologically sorted into the work list [15, pp. 612].

 ISBN: 978-1-61208-686-6 

**Input:** Q-graph $Q$ and selection operator $\sqcup$.
**Output:** For each $S \in \mathbb{S}(Q)$ the remaining items $rem(S)$.

```
/** Initialize **/
for all S ∈ 𝕊(Q) do
    rem(S) ← 0
worklist ← 𝕊(Q), visited ← ∅
/** Iterate **/
while worklist ≠ ∅ do
    S ← takeFirstOf(worklist)
    dirSucc ← {succ: (S, succ) ∈ S◁}
    if dirSucc ⊆ visited then
        if dirSucc = ∅ then
            rem(S) ← |S|
        else if |dirSucc| = |{succ}| = 1 then
            rem(S) ← |S| + rem(succ)
        else
            rem(S) ← |S| + ⨆_{succ∈dirSucc} rem(succ)
        visited ← visited ∪ {S}
    else
        putAtEnd(worklist, S)
```

Figure 3. General algorithm for remaining items

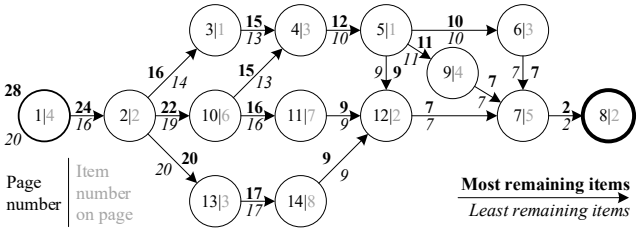


Figure 4. A sample Q-graph with page numbers (black), number of items on this page (grey), most (number above the edge) and least (number below the edge) remaining items.

## V. Example Strategies

To accentuate (4) and the algorithm of Figure 3 and to show how they can be firstly used to compare arbitrary strategies, they are applied on two example selection operators in this section. Both selection operators base on the proposed strategies of Kaczmirek [1][pp. 158-159]:

1) Select the most remaining items (longest case).
2) Select the least remaining items (shortest case).

### A. Most Remaining Items (Longest Case)

Taking a $\sqcup$ operator, which always selects the direct successor with the most remaining items, means that it chooses the *maximum* number of remaining items of all direct successors, $\sqcup = max$.

Figure 4 shows a Q-graph with a unique page index (the black number in the page) and the number of items on each page (the grey number in the page). Take page 5 of the Q-graph as an example. It has 1 item and three direct successors. It can be easily reconstructed that 10 items remain following the upper path $(S_5, S_6, S_7, S_8)$. On the path in the middle, there are 11 items, and, finally, on the lower path, there are only 9 remaining items. Since the maximum of all three paths is taken, we get $1+max(10, 11, 9) = 1+11 = 12$ remaining items for page 5 following Figure 3.

TABLE I
Progresses on path $(S_1, S_2, S_{13}, S_{14}, S_{12}, S_7, S_8)$ for the most, least and true remaining items.

| Progress | Most Items | Least Items | True Items |
|---|---|---|---|
| $\rho(S_1)$ | 14.29 % | 20.00 % | 15.38 % |
| $\rho(S_2)$ | 21.43 % | 30.00 % | 23.08 % |
| $\rho(S_{13})$ | 33.21 % | 40.50 % | 34.62 % |
| $\rho(S_{14})$ | 64.64 % | 68.50 % | 65.38 % |
| $\rho(S_{12})$ | 72.50 % | 75.50 % | 73.08 % |
| $\rho(S_7)$ | 92.14 % | 93.00 % | 92.31 % |
| $\rho(S_8)$ | 100.00 % | 100.00 % | 100.00 % |

Applying the algorithm of Figure 3 to the Q-graph initializes each page at the begin with a remaining number of items 0. Afterwards, it puts each page in the work list. For the sample Q-graph, the work list contains the following pages in a perfect order: $(S_8, S_7, S_6, S_9, S_{12}, S_5, S_4, S_3, S_{11}, S_{10}, S_{14}, S_{13}, S_2, S_1)$. In this order, each page only has to visited once, since when a page $S$ occurs, then all successor pages occurred before (this can be simply checked by comparing the order with the Q-graph). The algorithm computes the number of remaining items with $max$ as selection operator. This results in the number of most remaining items of Figure 4 (numbers *above* the edges), where this number for a page is annotated on its incoming edges.

If our dynamic equation (4) is used with these values, the progress on each page can be computed. Please note, the progress depends on the visited path. For example, if the path $(S_1, S_2, S_{13}, S_{14}, S_{12}, S_7, S_8)$ is taken by a participant, the resulting progress can be found in the second column, "*Most Items*", of Table I. If instead of page $S_{13}$ the participant goes to page $S_3$, $(\ldots, S_3, S_4, S_5, \ldots)$, the progress changes on page $S_3$ to $\rho(S_3) = 26.34\%$ instead of $\rho(S_{13}) = 33.21\%$ — a difference of more than $6\%$. Remember, the computed progress $\rho$ describes the progress *after* finishing the current page and, therefore, the progress at the start of the next page.

### B. Least Remaining Items (Shortest Case)

The determination of the *least* remaining items is similar to the determination of the most items although the selection operator is $min$. In this case, the values of remaining items of the Q-graph of Figure 4 change to the numbers *below* the edges in the same figure. The progresses for the path $(S_1, S_2, S_{13}, S_{14}, S_{12}, S_7, S_8)$ can be found in the third column, "*Least Items*", of Table I.

### C. Comparison of Both Strategies

In the last two subsections, it was explained, how the most and least remaining items strategies work. Although the same path $(S_1, S_2, S_{13}, S_{14}, S_{12}, S_7, S_8)$ of the Q-graph of Figure 4 was considered, the progresses on the pages diverge between both approaches (cf. Table I). But is it possible to decide, which of the both strategies is the better one?

As mentioned in the introduction, we need a trustworthy calculation of the progress [2, p. 757]. There is a lot of discussion if a computed progress is trustworthy or not.

 ISBN: 978-1-61208-686-6 

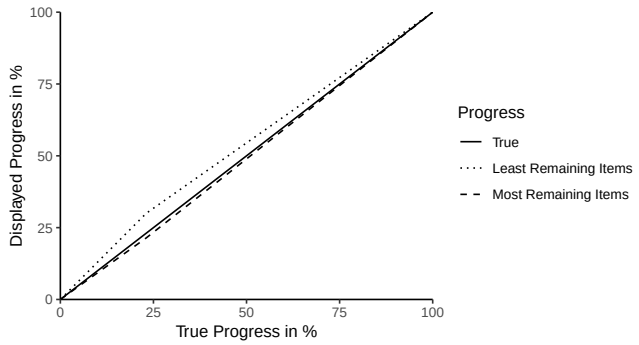


Figure 5. The *true* progress (black), the progress with least items (dotted line) and most items (dashed line)

We would argue that a computed progress is as trustworthy as it can be if the difference to the *true* progress is as small as possible. Remember, the *true* progress can only be determined *after* a participant has finished the survey, because at this point of time, we know exactly which path was taken and which was the total number of items. With this information, we can compute the *true* progress for each page easily by (2) for simple questionnaires.

Let us consider the *true* progress on each page of the previous example path. The total number of items is 26 on this path. Therefore, each item influences the progress by $1/26$ and the resulting progresses on the pages can be found in the last column, "*True Items*", of Table I.

The differences of the *true* progress on each page with the values of the computed progress with the most and least remaining items are visualized in Figure 5. The axes of the chart show the progress in percent.

The figure shows that on the least items strategy, the progress is *overestimated*, whereas on the most items approach, the progress is *underestimated*; both regarding the *true* progress. This is always true since the maximum number of remaining items is always greater or equal to the number of the true remaining items as well as the minimum number is always less or equal to the true number of items. Therefore, the progress with the least items strategy grows faster than the true progress at the beginning. The progress using the most items approach grows slower.

Considering the state of the art in Section II, the least items approach may be the better choice since there is a significant tendency in most studies that a fast-to-slow PI reduces break-offs. However, this was the consideration of one path of the Q-graph of Figure 4 only, and although the least remaining items approach fits to the hypothesis of the state of the art, it it is not sure that this approach is more trustworthy too.

As mentioned before, in our approach, a strategy (or selection operator $\sqcup$) is as trustworthy as possible if the difference between the computed progress to the *true* progress is as small as possible. In other words, the area between the computed and *true* progress should tend to be 0. In the chart of Figure 5, the area between the most items approach to the true progress is smaller than the area between the least items approach and the true progress, i. e., the most remaining items approach fits better to the *true* progress than the least remaining items approach. Remember, this is valid for the current considered path and should not be generalized. For an other path (e. g., the upper path in Figure 4), the least items approach could be the better choice.

As it was shown in this comparison, there are different arguments supporting both strategies. According to which survey and even path is given, either the most or least remaining strategy is more trustworthy. Furthermore, there could be other strategies, which fits better to a given survey than the two introduced one. Therefore, it is not easy to select the best item prediction strategy for a given survey. Furthermore, the proposed strategy of Kaczmirek [1][pp. 164-166] — taking the least remaining items — should be handled with care.

## VI. Conclusion

Including a progress indicator in a web survey seems to be a simple part of implementation. We have shown that this is only true for linear surveys and does not hold true any more for surveys with adaptivity (branches). This paper introduced a questionnaire model, which is named Q-graph. Based on this Q-graph, the difficulties of calculating the progress in Q-graphs with branches were shown. As a solution, we propose a general equation and algorithm to compute the progress dynamically based on the ideas of Kaczmirek [1]. The algorithm predicts the number of remaining items. Since this number depends on a prediction strategy, the algorithm needs a so-called selection operator as input. This makes it possible to compare different prediction strategies in the first place. Two examples of such operators were illustrated, which estimate the most or the least remaining items. The paper concluded with a comparison of both strategies and avenues for future research.

Since it is not trivial to choose the best strategy for a given survey, it is of interest to formulate an optimization problem for the selection of a best fitting selection operator for a given survey. This should be accentuated by a case study of different Q-graphs. It is also of interest to find other selection operators, which are, for example, more intelligent by using more information about the survey. One possible idea is to check conditions in our model and to determine the resulting path for a participant as early as possible.

## Appendix

*Theorem 1:* Let $Q = (\mathbb{S}, \mathbb{E})$ be a linear Q-graph. There is exact one path $(S_1, S_2, \ldots, S_m)$, $m \geq 1$, where $S_1$ is the start and $S_m$ is the end page. Let $I_i$ be the number of remaining items on page $S_i$. Since $Q$ is linear, this number of remaining items $I_i$ can be computed as follows:

$$rem(S_i) = I_i = I_{i-1} - |S_{i-1}| = I_1 - \sum_{j=1}^{i-1} |S_j| \quad (5)$$

In the case of a linear Q-graph, (4) is equal to (2):

$$\rho(S_i) \overset{(4)\ \&\ (5)}{=} \rho(S_{i-1}) + |S_i|\frac{1-\rho(S_{i-1})}{I_i} \tag{6}$$

$$=$$

$$\rho(S_i) \overset{(2)\ \&\ (5)}{=} \rho(S_{i-1}) + \frac{|S_i|}{I_1} \overset{(2)}{=} \frac{1}{I_1}\sum_{j=1}^{i}|S_j| \tag{7}$$

*Proof:* The proof is done by mathematical induction.
**Base:** Proof for $i = 1$ and $i = 2$:
$i = 1$:

$$\rho(S_1) \overset{(6)}{=} \rho(S_0) + |S_1|\frac{1-\rho(S_0)}{I_1} \overset{\rho(S_0)=0}{=} \frac{|S_1|}{I_1} \quad \checkmark \tag{8}$$

$i = 2$:

$$\rho(S_2) \overset{(6)}{=} \rho(S_1) + |S_2|\frac{1-\rho(S_1)}{I_2} \overset{(8)}{=} \rho(S_1) + |S_2|\frac{1-\frac{|S_1|}{I_1}}{I_1-|S_1|}$$

$$\overset{1=\frac{I_1}{I_1}}{=} \rho(S_1) + |S_2|\frac{\frac{I_1-|S_1|}{I_1}}{I_1-|S_1|} = \rho(S_1) + \frac{|S_2|}{I_1} \quad \checkmark$$

**Step case:** It is assumed that the theorem holds true for $i$. Proof for $i + 1$:

$$\rho(S_{i+1}) \overset{(6)}{=} \rho(S_i) + |S_{i+1}|\frac{1-\rho(S_i)}{I_{i+1}}$$

$$\overset{\text{theorem}}{=} \rho(S_i) + |S_{i+1}|\frac{1-\frac{1}{I_1}\sum_{j=1}^{i}|S_j|}{I_1-\sum_{j=1}^{i}|S_j|}$$

$$\overset{1=\frac{I_1}{I_1}}{=} \rho(S_i) + |S_{i+1}|\frac{\frac{I_1-\sum_{j=1}^{i}|S_j|}{I_1}}{I_1-\sum_{j=1}^{i}|S_j|}$$

$$= \rho(S_i) + \frac{|S_{i+1}|}{I_1} \quad \checkmark$$

■

## REFERENCES


[1] L. Kaczmirek, Human Survey-Interaction. Usability and Nonresponse in Online Surveys, 1st ed., ser. Neue Schriften zur Online-Forschung. Cologne, Germany: Herbert von Halem Verlag, 2009, no. 6.

[2] A. Villar, M. Callegaro, and Y. Yang, "Where Am I? A Meta-Analysis of Experiments on the Effects of Progress Indicators for Web Surveys," Social Science Computer Review, vol. 31, no. 6, 2013, pp. 744–762.

[3] SurveyMonkey Inc., "SurveyMonkey: Progress Bar," Website, available: https://help.surveymonkey.com/articles/en/kb/How-can-I-add-a-progress-bar-to-my-survey, retrieved: November, 2018.

[4] T. M. Prinz, R. Bernhardt, L. Gräfe, J. Plötner, and A. Vetterlein, "Using Service-oriented Architectures for Online Surveys in Coast," in Service Computation 2018: The Tenth International Conferences on Advanced Service Computing, Barcelona, Spain, February 18–22, 2018. Proceedings, pp. 1–4.

[5] T. M. Prinz, L. Gräfe, J. Plötner, and A. Vetterlein, "Statische Analysen von Online-Befragungen mit der Programmiersprache *liQuid* (Static Analysis of Online Surveys with the Help of the Programming Language *liQuid*)," in Proceedings 19. Kolloquium Programmiersprachen und Grundlagen der Programmierung, KPS, Weimar, Germany, pp. 59–70, September 25–27, 2017.

[6] B. A. Myers, "INCENSE: A System for Displaying Data Structures," SIGGRAPH Comput. Graph., vol. 17, no. 3, Jul. 1983, pp. 115–125.

[7] ——, "The Importance of Percent-done Progress Indicators for Computer-human Interfaces," SIGCHI Bull., vol. 16, no. 4, Apr. 1985, pp. 11–17.

[8] C. Harrison, B. Amento, S. Kuznetsov, and R. Bell, "Rethinking the Progress Bar," in Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, 2007, pp. 115–118.

[9] D. A. Dillman, R. D. Tortora, and D. Bowker, "Principles for constructing web surveys," Social and Economic Sciences Research Center (SESRC), Washington State University, Pullman, Washington, USA, Technical Report 98-50, 1998.

[10] F. G. Conrad, M. P. Couper, and R. Tourangeau, "Effectiveness of progress indicators in web surveys: It's what's up front that counts," in Survey and statistical computing IV. The impact of technology on the survey process, pp. 1 – 10.

[11] U. Matzat, C. Snijders, and W. van der Horst, "Effects of Different Types of Progress Indicators on Drop-Out Rates in Web Surveys," Social Psychology, no. 40, 2009, pp. 43–52.

[12] F. G. Conrad, M. P. Couper, R. Tourangeau, and A. Peytchev, "The impact of progress indicators on task completion," Interacting with computers, no. 5, 2010, pp. 417—427.

[13] M. Liu and L. Wronski, "Examining completion rates in web surveys via over 25,000 real-world surveys," Social Science Computer Review, vol. 36, no. 1, 2018, pp. 116–124.

[14] G. Chartrand and P. Zhang, Discrete Mathematics, ser. 1 edn. Long Grove, Illinois, USA: Waveland Press, Inc., 2011.

[15] T. H. Cormen, C. E. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, ser. 3. Cambridge, UK: PHI Learning, 2010.

[16] P. J. Pahl and R. Damrath, Mathematical Foundations of Computational Engineering: A Handbook, ser. 1. Auflage, F. Pahl, Ed. Berlin, Germany: Springer, 2001.

[17] J. Rost, Lehrbuch Testtheorie – Testkonstruktion (Textbook Test Theory – Test Construction), ser. Zweite, vollständig überarbeitete und erweiterte Auflage. Bern, Switzerland: Huber, 2004.

[18] H. Moosbrugger and A. Kelava, Eds., Testtheorie und Fragebogenkonstruktion (Test Theory and Survey Construction), ser. 2., aktualisierte und überarbeitete Auflage. Berlin, Germany: Springer, 2011.


 ISBN: 978-1-61208-686-6 