Latent program modeling: Inferring latent problem-solving strategies from a PISA problem-solving task

Erik Lundgren
Umeå University
erik.lundgren01@umu.se

Response process data have the potential to provide a rich description of test-takers' thinking processes. However, retrieving insights from these data presents a challenge for educational assessments and educational data mining as they are complex and not well annotated. The present study addresses this challenge by developing a computational model that simulates how different problem-solving strategies would behave while searching for a solution to a Program for International Student Assessment (PISA) 2012 problem-solving item, and uses n-gram processing of data together with a naïve Bayesian classifier to infer latent problem-solving strategies from the test-takers' response process data. The retrieval of simulated strategies improved with increased n-gram length, reaching an accuracy of 0.72 on the original PISA task. Applying the model to generalized versions of the task showed that classification accuracy increased with problem size and the mean number of actions, reaching a classification accuracy of 0.90 for certain task versions. The strategy that was most efficient and effective in the PISA Traffic task evaluated paths based on the labeled travel time. However, in generalized versions of the task, a straight line strategy was more effective. When applying the classifier to empirical data, most test-takers were classified as using a random path strategy (46%). Test-takers classified as using the travel time strategy had the highest probability of solving the task $(\hat{p} \approx 1)$. The test-takers classified as using the random actions strategy had the lowest probability of solving the task $(\hat{p} \approx 0.11)$. The effect of (classified) strategy on general PISA problem-solving performance was overall weak, except for a negative effect for the random actions strategy $(\beta \approx -65, CI_{95\%} \approx [-96, -36])$. The study contributes with a novel approach to inferring latent problem-solving strategies from action sequences. The study also illustrates how simulations can provide valuable information about item design by exploring how changing item properties could affect the accuracy of inferences about unobserved problem-solving strategies.

Keywords: process data, computational cognitive modeling, PISA, problem-solving, educational assessment

1. Introduction

The goal of educational data mining is to use data mining techniques to improve education (Romero and Ventura, 2020). One aspect of education is the assessment of skills and knowledge of learners, from the individual level to the societal level. The response processes of test-takers have a given place in the validation of test scores and are emphasized in the Standards for

Educational and Psychological Testing (2014). However, this rich source of information is rarely used in research and practice (Ercikan and Pellegrino, 2017; Hubley and Zumbo, 2017). With the advent of computer-administered assessments and complex interactive test items, response process data have become widely available, making the development of research methods on how to best extract information from response process data increasingly relevant. The usual observational outcomes from large-scale educational tests are *product data*, a final result of a problem-solving process that is often coded as 1 or 0 indicating if a test item was solved or not solved. In comparison, *response processes data* refer to data of comparatively greater resolution, which provides additional information concerning intermediate events or measurements that preceded the product data, however, the distinction between product- and process data can be fuzzy as noted by Levy (2020). Process data are messy and abundant, one item could contain as many observations as there are items in an assessment. This makes educational process data a very interesting topic for educational data mining. Embedded in the noisy process data are interesting clues regarding what kind of thinking processes could have caused test-takers to behave in the way they did. The increased amount of behavioral measurements together with innovative item designs make it difficult to imagine, evaluate and predict all the different ways that it is possible to respond to tasks, and what the responses reveal about test-takers regarding the purpose of the assessment. That is, the interpretation of response processes as indicators of latent constructs is not always straightforward. This is partly because the cognitive models in most assessments, particularly large-scale assessments, are described at a high level of test specification or domain mastery, and not at the level of *task performance* (Leighton and Gierl, 2007).

Data mining methods within the educational context fall within five general categories (Baker, 2010). The first three, prediction, clustering, and relationship mining, are general to all areas of data mining, while the remaining two, discovery with models, and distillation of data for human judgment, are of specific interest in educational data mining. An example of a prediction and clustering study applied to process data from educational assessments include Qiao and Jiao (2018) which explored both supervised and unsupervised data mining techniques to analyze response process data from PISA 2012 problem-solving items and found that they could predict score labels from process data variables with high accuracy. Another example is Salles et al. (2020) which used different clustering techniques to extract problem-solving strategies from educational log data, but could not find that their unsupervised technique provided a satisfactory explanation of why students achieved the correct answer to the test item. Greiff et al. (2015) explored how an a priori developed process-data derived indicator of a specified problem-solving strategy (isolated variation of variables) was related to task performance on a PISA problem-solving task and found that it related to high percentages of correctly solving the item, while the absence of using the strategy was related to low percent of solving the task. The importance of the strategy was corroborated by an unsupervised approach (latent class analysis) in a later study (Xu et al., 2018). Other approaches used to analyze educational process data include latent topic models (Xu et al., 2020), autoencoders (Tang et al., 2021), and linear dynamic systems (Paassen et al., 2021).

The present research takes an approach that differs somewhat from the pure data mining techniques by using a simple data mining technique in combination with a stochastic simulation model. The simulation model represents the data generating process of different problem-solving strategies, and the data mining technique is applied to infer which problem-solving strategy most probably generated an observed action sequence. And as such it could be char-

acterized as a kind of *discovery with models* since the result from one data mining analysis is used in another analysis (Baker and Siemens, 2014). Computational models are characterized by a close conceptual relationship between a theory of a latent construct and a model of the construct, and aim at both explaining and generating behavior, which is useful in many ways. For example, they could provide an understanding of potential reasons why test-takers failed or passed a task, an understanding of what kind of thinking was required to solve the task, and what kind of behaviors could be generated by different ways of thinking.

Several areas have been suggested in which computational cognitive models are relevant to assessment and education. For example, automated assessment and instruction (Rafferty, 2014), understanding data from complex tasks (LaMar, 2014), and predict skill decay and personalizing training for raters of constructed response scoring (Walsh et al., 2021). They could also be used for providing validity evidence regarding the interpretation of scores (Kane and Mislevy, 2017). A report by (Moon et al., 2018) suggested that computational cognitive models could be used to evaluate item format designs, provide validity evidence and automated identification of solution strategies, and improve human scoring.

Much of the focus of Artificial Intelligence (AI) and Machine Learning (ML) methods is on prediction, but greater insight into the explanatory power of models has been asked for (Rosé et al., 2019), something which computational models could help address. Researchers have also asked for item designs that allow for the clear identification of different strategies (He et al., 2021). By simulating problem-solving behavior from computational models, it is possible to address the issue of strategy identification, for example by modeling how changes in item design would affect the accuracy of inferences about latent strategies.

The present study seeks to contribute to the ongoing research on process data in educational assessment and data mining by using computational cognitive modeling together with a basic data mining method of text classification to make inferences from observed process data to latent problem-solving strategies.

1.1. Aim

The aim of this study is to:

1. Develop a computational model that explains and simulates how the observed problem-solving behaviors could be generated using different problem-solving strategies in a PISA problem-solving task.

2. Investigate to what extent latent problem-solving strategies can be retrieved from the observed behaviors (2a), and whether the accuracy of retrieval is affected by changing the item design (2b).

3. Assess the performance of different problem-solving strategies in the original version of the PISA problem-solving task (3a) and in generalized versions of the task (3b).

4. Achieve further evidence of validity regarding the latent strategy inference by investigating how the test-takers' predicted strategies relate to task performance (4a) and overall problem-solving competency as measured by PISA (4b).

1.2. Organization of study

The study proceeds by introducing the PISA problem-solving item *Traffic* in Section 2. This is followed by Section 3 with a brief theoretical background on problem-solving. The computational cognitive model representing problem-solving in the Traffic task is presented in Section 4. Section 5 outlines the methods used to infer the latent strategies. Section 6 presents three simulation studies that assess the correspondence between the model and empirical behavior, the retrieval accuracy of latent strategy inference, and the problem-solving performance of the different strategies. In Section 7, latent strategy inference is applied to empirical data from the PISA test-takers to achieve further validity evidence and investigate how the inferred strategy of real test-takers relates to task performance and general PISA performance. Finally, the study ends with a discussion in Section 8.

2. PISA and the Traffic task

Programme for International Student Assessment (PISA) is a triennial large-scale international comparative study that assesses 15-year-old students' in the domains of mathematics, science, and reading (OECD, 2013). In 2012, 65 countries and economies participated in PISA (OECD, 2014). In addition to the regular PISA assessment battery, PISA 2012 also featured an optional computer-based problem-solving assessment, and about 85 000 students in 44 countries took part in this assessment. The present study used data from all students that were administered the Traffic task (Figure 1) as part of the problem-solving assessment, around 25000 students from all participating countries. The present study focuses on Traffic item 2 (coded CP07Q02), which can be viewed and tested at `www.oecd.org/pisa/test-2012/testquestions/question2/`. The task comprises stimuli depicting a road network with associated travel times between cities. The test-takers were introduced to the task with the following written prompt: "Maria wants to travel from Diamond to Einstein. The quickest route takes 31 minutes. Highlight this route." (OECD, 2014, p.41, Figure v.1.17). The test-takers could highlight and unhighlight routes by clicking on sections of paths (the *edges* in graph-theoretical terms) and remove their selected path with a reset button. When a section was highlighted, the total travel time of the selected paths was added together and shown in a box in the lower-left region of the stimuli. The Traffic task was part of a set of items constructed to measure a construct called "problem-solving competency" which comprised: exploring and understanding, representing and formulating, planning and executing, and monitoring and reflecting. The Traffic task was specifically designed to measure the sub-construct *planning and executing*: "planning, which consists of goal setting, including clarifying the overall goal, and setting subgoals, where necessary; and devising a plan or strategy to reach the goal state, including the steps to be undertaken" (OECD, 2014, p.82), and "executing, which consists of carrying out a plan" (OECD, 2014, p.83). Specifically for question 2 of the Traffic task, we can read that "Students can use the indication that the quickest route takes 31 minutes to avoid generating all possible alternatives systematically; instead, they can explore the network in a targeted way to find the route that takes 31 minutes" (OECD, 2014, p.41).

According to the framework outlined by PISA, the Traffic task requires test-takers to set goals, formulate a plan or strategy that explores the network in a targeted way, and execute the strategy. The Traffic task was scored as product data, with a 1 if the correct path was highlighted when ending the task, and with a 0 if the correct path was not highlighted. From the assessment's

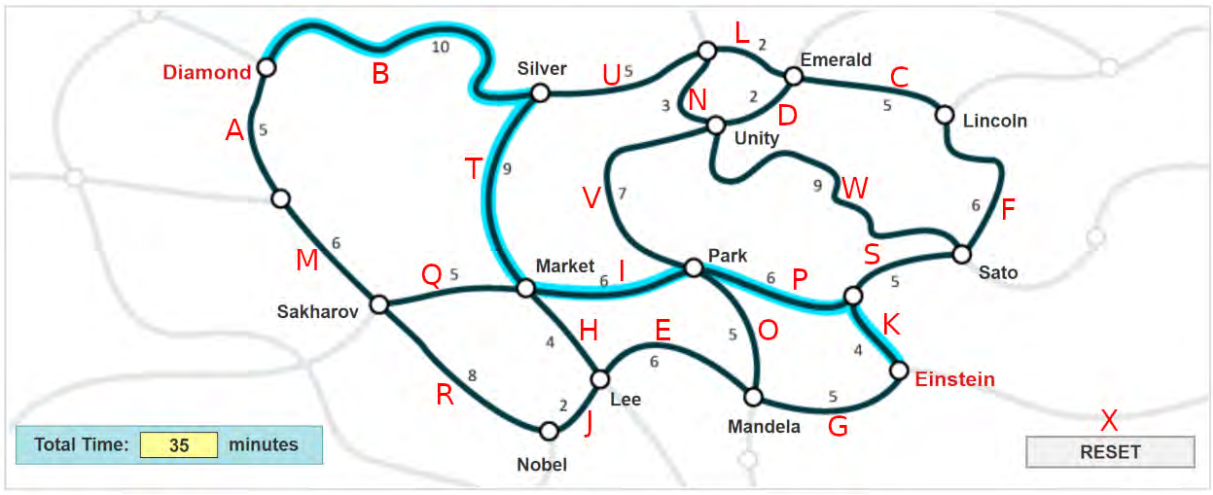


Figure 1: Traffic task. Labels of action events are shown in capital letters, B refers to a click on the link between Diamond to Silver, X to a click on the reset button, and so on. Using the notation, we can say that the path B, T, I, P, K has been highlighted.

perspective, this information is sufficient to conclude that the test-taker engaged in the cognitive processes of planning and execution as defined by PISA and that a score of 1 indicates a higher ability on the latent construct compared to a score of 0. Can we, however, be sure that test-takers that solved the task were better at planning and formulating a strategy, as well as executing it? It can be assumed that different strategies can be followed in order to explore the road network and find the path that solves the task. Some strategies might search in a "targeted way" and use different kinds of information in the planning of actions, while other strategies perhaps rely on less targeted exploration of random trial and error. While product data do not give any information that could help differentiate between such strategies, perhaps process data do. However, this reasoning requires a link between the behavioral responses and the latent construct of interest. The present study creates such a link by creating a computer program that models how problem-solving behavior in the Traffic task could be generated by following different latent problem-solving strategies.

3. PROBLEM-SOLVING STRATEGIES

The overall perspective on problem solving used in this study comes from Newell (1990) which considers problem solving as a search process over a problem space. A problem-solving agent faces a problem when the agent is in a state, wants to be in another state (a goal state), but does not know what to do in order to transfer from the current state to the goal state. Since the agent does not immediately know what to do, it has to search among the available actions. The problem-solving agent can use available knowledge to narrow down the search by prioritizing alternative actions. If the agent has complete knowledge of the problem, it does not have to search for a solution, it can just retrieve the solution from memory, execute the necessary actions, and arrive at the goal state. In planning, the agent tries to solve (parts of) the problem before executing a solution. Planning can be carried out *offline* by completely planning a solution before taking any action in the environment, or *online* by planning in real-time while acting in

the environment. In the present study, no sharp distinction is made between problem solving and planning, since planning can be understood as a kind of problem-solving task in itself. In the Traffic task, we can think of the planning as being very similar to a problem-solving strategy. A problem solver has the overarching goal of finding the shortest path, and some kind of strategy or plan has to be followed in order to come up with a set of actions that hopefully will select the path that solves the problem. Once the plan, or parts of the plan, are formulated, the plan is tested by executing actions and observing the result. The following subsection provides some clues regarding factors that could be taken into consideration by problem solvers when creating plans to try to solve the Traffic task.

3.1. Problem solving in shortest path search and related problems

There exists a rich library of algorithms and approaches within the field of computer science with respect to state-space search and path planning. Some basic search algorithms for searching a problem graph are breadth-first search (BFS), depth-first search (DFS), Dijkstra's algorithm, greedy best-first search, recursive best-first search (RBFS), and the A* algorithm (Edelkamp and Schrodl, 2011; Russell and Norvig, 2021). BFS explores all unexplored neighboring nodes in the problem tree or graph that can be reached from the starting level before proceeding to explore nodes in the next level. DFS explores as far down it can until it hits a dead end, it then backs up levels to the first unexplored node and starts exploring further down again. Dijkstra's algorithm is similar to BFS, but return optimal solutions to single-source shortest path problem on graphs that have weighted edges. Dijkstra's algorithm works by keeping track of the minimum distance from a predecessor node to the next node. Algorithms BFS, DFS, and Djikstra are *uninformed* algorithms, meaning that they do not take any additional estimates into account in their node selection than the information given from a neighboring node. A* algorithm differs from the previous algorithms by being guided by a *heuristic*. A heuristic is some evaluation of a future state, or an estimate of a cost to reach a goal state, which makes it more "promising" to move to certain nodes. A* is similar to Djikstra's algorithm but guides the search by adding a heuristic value to the edge weight costs. Greedy best-first search algorithms simply decide to move to the node with the best heuristic value.

Contrary to the imaginative, creative, and satisficing (non-optimal) nature of human problem solving, the basic algorithms from computer science mentioned above are developed to return solutions, some of which can be guaranteed to return optimal solutions under certain circumstances. The basic computer science algorithms execute their actions deterministically and do not make mistakes or forget which paths and solutions they have previously explored (if not explicitly told to do so). Standard AI algorithms like A* produce fixed paths without variation, something that goes against what is characteristic of human path planning (Dubey et al., 2022). A* also requires a lot of memory since it needs to keep track of all previously reached states. Thus, many basic computer science algorithms are not, in their original form, good candidates for models over human behavior, they can however be modified into more realistic models for human thinking and behavior. Rahmani and Pelechano (2022) proposed a DFS algorithm that was modified with a greedy heuristic to model human search in unknown environments. The heuristic was based on orientation toward the goal since it has been reported that humans like to walk as far as they can in a straight line toward goal locations. DFS was also chosen since humans are characterized to explore sequentially towards the goal. However, the more knowledge humans had of the environment the more their solution path lengths started to approach

solutions with optimal path lengths as given by an A* search algorithm.

The problem that is of focus in this study, the PISA Traffic task, is a task where the goal is to find and highlight the shortest path between two nodes in a weighted 2D graph where the weights are explicitly labeled next to the edges, and where the shortest path is indicated by an exact total travel time that the problem solver already knows from the start. Only a few studies could be found that have investigated which strategies humans rely on to solve shortest path problems in 2D graphs, and no studies could be found that explore human problem-solving strategies on problems that are identical to the PISA Traffic task. On shortest path problems on 2D graphs without weights, Huang et al. (2009) observed a tendency among the participants to prioritize paths based on how close they are to a hypothetical *straight-line path* between the start and target nodes, results that are in line with (Dalton, 2003). The preference for straight-line paths is related to another human preference to select paths based on minimizing the angles in paths or towards the goal (Turner, 2009; Hochmair and Frank, 2000). Factors that influence the difficulty of finding the shortest path in graphs are the number of edges in a path, path continuity, the angular deviation of adjacent edges on a path, the number of edge crossings, and the number of nodes in the path (Ware et al., 2002). Unsurprisingly path planning problems become more difficult with increased size or complexity (Reitter and Lebiere, 2010; Dye, 2007).

An eye-tracking study by Netzel et al. (2014) investigated reading strategies on metro maps. The task was to find a route between a start and target station and state how many transitions were necessary. Different clusters of eye movements were identified and interpreted as indicating three different strategies. Some of the participants seemed to start the search by foveating the start node and then incrementally and locally moving closer to the goal node. Another strategy involved the participants starting the search by jumping from the start node to the goal node and then back to the start node before starting to work their way locally from the start node to the goal node. In yet another strategy, the participants made serial fixations back and forth between the start and goal nodes, probably verifying the path.

A problem related to shortest path-problems is the traveling salesperson problem (TSP). In a TSP the goal is to find the shortest round trip that visits all nodes on a graph. Hypothesized strategies proposed for this kind of task have been a local nearest-neighbor strategy (Gärling, 1989), predominately so when only numerical distances are presented to problem solvers, but also when a spatial picture representation is presented. Other proposed strategies include a sequential convex hull model in which the participants start by connecting the outer boundary points, strategies relying on cross avoidance, and strategies using global-to-local processing (MacGregor and Chu, 2011). How far problem solvers can look into future states of the problem depend on how far they can keep a *mental lookahead*. According to Basso et al. (2001) the participants only plan a small section at a time and then readjust their planning as they progress. Reitter and Lebiere (2010) proposed that humans search by using an egocentric strategy when planning paths in 2D maps due to limits of visual attention and memory and that these limitations lead to a planning process with local increments and local optimization. Similarly, a study comparing computational models of a search process in a real-life search task found that realistic search involves local decisions and goal-directed behavior (Mueller et al., 2013); in this study a goal scent model with multiple goals showed the best fit with respect to human-produced data, although a model using random goals did also replicate human data well in many aspects. A similar, locally initiated, fine-to-coarse hierarchical path planning strategy has been proposed by Wiener and Mallot (2003). In this strategy, the planning agent uses fine detail information for close surroundings and less detailed information for distant spaces to plan a route to the local

region where the goal is located, after which a path is planned to the exact location of the goal.

Previous studies have not used tasks that are identical to the PISA Traffic task. However, they are still relevant regarding possible visual heuristic strategies for finding the shortest paths and the interpretation of the results in the present study. In summary, it would appear that when humans use visual heuristics on unlabeled graphs, they use heuristics that favor short and straight paths, that try to avoid sharp angles and crossings. Problems become more difficult as they become larger in size or more complex. Problem solvers seem to be limited in how much they can plan ahead of time, which can lead to non-optimal solutions. Previous research indicates that a variety of strategies exist, strategies that rely more on local processing, other strategies that explore more globally, and strategies that are a combination of both.

3.2. Problem-solving strategies in the PISA Traffic task

Process data from the test-takers' behaviors on the PISA 2012 Traffic task have been the subject of a few previous studies. Liu et al. (2018) used the Traffic task to apply a modified multi-level mixture IRT model. They found that the model could both differentiate between response strategies and provide estimates of ability. The strategy classes were based on the behavior of selecting identical or similar routes multiple times. They found no differences between the group that solved the task and the group that did not solve the task regarding the frequencies of selecting wrong paths.

Fang and Ying (2020) used the first item from the Traffic task scenario (Code CP07Q01) to apply a latent theme dictionary model for identifying patterns in the process data. The method reduces the test-takers' sequences to sub-sequences (sentences) by a cutting process defined subjectively, or by special events which in the case of the Traffic task be actions of resetting, highlighting, or unhighlighting. The results showed that, on average, the test-takers had around 10 sentences (i.e. 10 changes in unhighlighting or highlighting). They found that some classes of test-takers had a higher probability of finding the correct answer, some were slower, some were more efficient, while other classes of test-takers did not use frequently observed action sequences.

Chen et al. (2022) analyzed the Traffic task with a Latent Space Model which they applied to cluster actions, error analysis of test-taker behavior, and as a performance measurement. They used an average linkage measurement in which larger values indicate less efficient problem solving, and lower values more efficient problem solving (if the item was solved). Test-takers who solved the task had generally lower average linkage values compared to test-takers who did not solve the task, however, the results also suggested variation in average linkage measurement for test-takers, which indicates that test-takers can solve and fail to solve the task with many different kinds of behaviors of different efficiency.

Lundgren and Eklöf (2020) used the Traffic task (CP07Q02) to explore test-takers' test-taking effort and motivation. The results showed that there was variation in the number of trials and actions undertaken by the test-takers before finding the solution. This suggests that different strategies guide the search. The results also showed that the Traffic task can apparently be solved by offline planning, that is, by not executing any observed actions initially and then delivering the correct solution. However, the predominant way of solving the task seemed to be by online planning.

In summary, previous research on the Traffic task suggests that there is a lot of variation in how the test-takers solve the tasks and that there are more and less efficient ways of solving

the tasks. The fact that it is common for the test-takers to try multiple paths before solving a task suggests an online trial and error search process, since had the solutions been completely planned before any actions were executed, the tester-taker would probably have only needed one attempt to solve the task. This, in turn, means that for most of the test-takers, the action sequences will leave information that reflects their cognitive search process and provide data that is informative of their latent problem-solving strategies, and as such the log-file data from the PISA 2012 problem-solving assessment make for a good case study to investigate how some of the previously mentioned assessment-relevant applications could be realized.

4. COMPUTATIONAL MODEL

4.1. REPRESENTATION OF THE TRAFFIC TASK PROBLEM

The Traffic task can be represented as a graph where the cities are represented by nodes, the roads connecting the cities are represented by edges, and the travel time of the roads is indicated by weights related to the edges. The set of possible actions includes actions that change the environment: clicking on any of the sections of roads and clicking on the reset button. If an unhighlighted edge is clicked, it is highlighted, and vice versa. If the reset button is clicked, any currently highlighted edge will become unhighlighted. The goal state of the task is to find a path between two cities that satisfies a predefined time cost.

4.2. MODEL OF PROBLEM-SOLVING IN TRAFFIC TASKS

A model is often developed from a verbal account of the data generating process. It was hypothesized that the way test-takers would try to solve the Traffic task is to start by visually exploring the stimuli, and locating and reading the instructions. Once the task is understood and the test-taker intends to try to solve the task, the test-taker will follow some strategy to search for a path that the test-taker believes will achieve the goal state of the task. The test-taker will then highlight the path by performing a series of clicks. Adjustments may be made along the way depending on how far the path has been planned. There might be some errors made in the process of the planning, or while executing the clicks. Having completed a path, the test-taker will be interested in learning whether the selected path corresponded to the correct solution to the problem. This is achieved by ensuring that there is a complete path between the target and goal location and that this path does not exceed the predefined time constraint. If the selected path was the correct path, the problem solver will have found the goal state, be satisfied with the performance, and move on to other tasks. If the selected path did not solve the problem, then the test-taker would try additional paths in the same way as described above, for as long as the test-taker is motivated to solve the task, or until the solution is found.

Algorithm 1 provides a high-level description of the problem-solving loop that was used to simulate the problem-solving behavior described above. In line 1, the problem solver starts by believing that the goal state has not been found. Line 2 starts the problem-solving loop that will run as long as the goal state has not been found and the test-taker is motivated to keep on trying. In line 3, the planning is done by a function `plan(h)` that takes argument h which is parameterized with the heuristic evaluation function that evaluates the cost of taking actions. The problem solver wants to minimize the cost of the heuristic function, prioritizing actions that lead to lower costs, see Algorithm 2 for a high-level description of the planning subroutine. The evaluation function represents one of six problem-solving strategies that are

Algorithm 1 Main loop of the problem-solving model

```
1: goal_state_found ← false
2: while (not goal_state_found) and motivated do
3:     planned_actions ← plan(h = strategy)
4:     for action in planned_actions do
5:         task.event(action)
6:     end for
7:     goal_state_found ← check_goal(task)
8: end while
```

explained in more detail in Section 4.3 below. Thus, we can think of line 3 as representing a planning phase where a test-taker plans their actions, to some depth and according to some strategy. The planning function returns a list of planned actions (`planned_actions`) that are executed one by one within the for-loop at line 5, where the function `task.event()` changes the configuration of the Traffic task environment depending on which action that was used as input, simulating a click on the stimuli. In line 7, the problem solver uses the function `check_goal()` to determine whether the correct path has been found. This simply simulates that a test-taker verifies whether the goal state criteria were fulfilled, making sure that there is a highlighted route between Diamond and Einstein with a travel time of exactly 31 minutes. If the goal state check returns `True` the while loop will terminate. Otherwise, it will repeat the process, provided that the problem solver is still motivated.

Algorithm 2 High-level description of planning subroutine

```
1: function plan(h)
2:     prioritize all sub-paths from the current node to depth d with cost from heuristic h
3:     return list of actions that completes the sub-path with the least heuristic cost
4: end function
```

It should be noted that this model produces the same kind of data as the empirical data (disregarding the information on time intervals between actions), that is ordered records of executed actions $(\text{action}_0, \text{action}_1, \ldots \text{action}_n)$, where each action is an action from the set of possible actions. The model also generates behavioral data sequentially, which is how real test-takers would generate it. The action sequences are dependent on which strategy is used by the agent, just as we believe that the sequence generated by a real test-taker is the result of the strategy that they used when trying to solve the task. We now move on to describe the strategies that are proposed as explaining the problem-solving behavior in the Traffic task.

4.3. Problem-solving strategies

Two uninformed and four heuristic strategies were considered. Uninformed strategies do not use any heuristic estimates to guide selection between possible actions, while the heuristic strategies guide their action selection by an evaluation function. See Russell and Norvig (2021) and Edelkamp and Schrodl (2011) for further details on heuristics and AI search algorithms.

Strategies 3 to 6 build on a combination of depth-first search (DFS), depth-limited search, best-first search, and iterative deepening search. The planning process works by first exploring depth-first to a predefined maximum depth limit, and then choosing the path with the lowest

heuristic cost (as such it is similar to the algorithm proposed Rahmani and Pelechano (2022) for human path-finding in unknown environments). If the depth limit is low, a complete path will not be planned between the start and goal node. In this case, the actions of the partial path will be executed once the depth limit is reached, and then another depth-limited search will depart from the end node of the first partial path. This type of search process was selected to model human problem-solving strategies since it follows the iterative exploratory behavior observed by the eye-tracking study (Netzel et al., 2014), and it aligns with the hierarchical path planning hypothesis (Wiener and Mallot, 2003). Iterative deepening DFS was chosen since it is related to *the progressive deepening* (de Groot, 1978) search strategy used by chess players, and endorsed by Newell and Simon (1972) as a model for general human problem solving due to limited working memory and short-term memory. The depth limit of the search process represents the mental lookahead of the modeled problem solver - how far down the road the agent can plan before deciding on following a path. This parameter can be varied to allow for differences in mental lookahead. A shallow depth leads to a local strategy and a greater depth to a more globally informed strategy. If the depth is set to 1, a decision is made after having only considered the nearest neighboring nodes, only looking one step ahead. A small lookahead could explain the error of taking a globally non-optimal path, running into dead ends, as observed by Huang et al. (2009). By increasing the search depth, it is possible to model a problem solver with a greater mental lookahead that can plan and compare paths of greater length.

4.3.1. Random actions

One very simple strategy is to close your eyes and click randomly on the edges or the reset button, and just hope that you solve the problem. Finding a solution with this strategy is for most problems as likely as winning the jackpot in the lottery. This strategy is unlikely to be used by any real problem solver but is it is included to represent the behavior of someone who interacts with the task but is not interested in trying to solve it, or a kind of residual class representing motivated problem-solving behavior that deviates from any of the other strategies and thus not well understood. The strategy this model uses to select actions was implemented by sampling actions from a uniform distribution over the set of available actions.

4.3.2. Random paths

A more reasonable strategy is to randomly try paths between the start and goal destinations. Try one path; if it is not the correct path, try another path, and so on, until you eventually find the path that solves the problem. This problem-solving strategy is very similar to depth-first search, but it is not constrained to explore nodes in a systematic order as depth-first does. This strategy was implemented by choosing actions arbitrarily as long as they were limited to creating paths that are plausible. ”Plausible” in this context means that the problem solver disregards extremely meandering paths, which were regulated by a threshold on the angular deviation of an edge with respect to the goal node. Note that this strategy is not strictly uninformed since it uses a constraint (based on the angle) on which paths it can choose from, it does not however use any heuristic to prioritize between paths that respect the constraint. However, if a test-taker was given an angle constraint equal to π (no constraint) then the forward phase of the algorithm would be a DFS search.

Table 1: Parameter setting in model and strategies.

Parameter	Value/function	Strategy
Motivation	Negative-Binomial($\lambda = 100, \kappa = 5$)	All
Start node	$P(Diamond) = 0.9$, $P(Einstein) = 0.1$	$\star$
Path switch method	Uniform($\{reset, backtrack, optimal, memory\}$)	$\star$
Angle threshold	Beta$(0.01, 1)$, transformed to interval $[\pi/2, \pi]$	$\star$
Search depth	Binomial$(5, 0.5) + 1$	$\triangle$
P(forgetting path)	0.1	$\star$
P(mistake)	0.1	$\star$
P(indecisiveness)	0.1	$\star$

$\star$ Not used by random actions; $\triangle$: only used by heuristic strategies

4.3.3. Travel time

The travel time strategy selects actions leading to the lowest travel time cost. Travel time is indicated by the labels near the edges. This model reflects an informed search process that focuses on minimizing what matters to solve the task: the travel times. This would be like comparing the travel times of all paths to a limited depth. This strategy is similar to Dijkstra's algorithm, but only searches for a limited depth.

4.3.4. Visual length

The visual length strategy selects actions by minimizing the visual length of paths. Visual lengths of paths were approximated by measuring the length in pixels of the road sections in Traffic task stimuli.

4.3.5. Straight line

The straight line strategy selects actions by minimizing the Euclidean distance between a neighboring node and an imagined straight line between the start and goal node. This strategy represents the straight line tendency observed by Huang et al. (2009).

4.3.6. Least angle

The least angle strategy evaluates actions by choosing the action that leads to the smallest angular deviation to the next future node compared to the alternative nodes. It is a sort of variation on the straight-line strategy but it does not necessarily try to select nodes that are close to a global straight line to the goal; it just tries to avoid sharp turns. This strategy is motivated by the observations in previous research that human problem solvers solving the TSP tend to avoid crossings, paths with sharp angular deviations, and follow the convex hull.

4.3.7. Parameter settings of strategies

All of the strategies (except random actions) have many common rules and parameters. In the empirical data, it was observed that most participants ($\approx 73\%$) first click was on edges connected to Diamond, suggesting that they used Diamond as their starting node to search for solutions. However, some participants ($\approx 8\%$) also started building paths from Einstein (the

remaining percentage started with some other action). For simplicity, since the models can only start building paths from Diamond or Einstein, and since 73% to 8% has a similar ratio to 90 vs 10, simulations were set to have a 0.9 probability to start from Diamond and 0.1 probability to start from Einstein. Since no loopy paths were observed to be created by test-takers, simulations were not allowed to create loopy paths (except for random actions). Out of the 95 non-loopy paths between Diamond and Einstein, there are many snake-like paths that were not considered by the test-takers. To address this, *most* problem-solvers were assigned an *angle threshold* value which helped them avoid selecting edges with very large angular deviations from the goal node. These angle thresholds were, for each simulated test-taker, sampled from a $\mathrm{Beta}(0.01, 1)$ scaled to $\pi/2$ to π radians. This leads the majority of simulated test-takers to avoid angles that are larger than 90 degrees while allowing some simulated test-takers very liberal angle thresholds leading them to explore zigzagged paths that account for outlier behavior.

To account for errors made when clicking, or by imperfect adherence to the strategy, simulated test-takers were set to have a probability of 0.1 of making a *mistake*, or an unintended, action, such that a neighboring edge was clicked instead of the one that was prioritized by the heuristic.

Real test-takers' memory is not perfect, this was modeled by adding a parameter that regulated the probability of forgetting a path at any cycle of planning selection, the *forgetting* parameter was set to 0.1. Some *indecisiveness* was observed in the data from test-takers, indicated by highlighting an edge, and then unhighlighting it, and then highlighting it again. This noise was modeled with a parameter that regulated the probability of making such "on-off" actions, it was set to 0.1.

For the strategies that use heuristic evaluations, variation in the mental lookahead was accounted for by varying the search depth limit. The search depths were generated by a $\mathrm{Binomial}(n = 5, p = 0.5) + 1$. Starting from Diamond, a search depth of 1 would require a comparison to be made between 2 paths in the working memory, a depth of 2 would require comparing 3 to 4 paths, a depth of 3 could require a comparison of 6 paths, and a depth of 4 would require even more comparisons, and so on. Thus, depths around 1 to 6 should approximately be within the limits of human working memory capacity. Working memory has often been considered to be limited to 7 ± 2. However, others studies have estimated it to be approximately 4 (Cowan, 2001). Thus, this parameter could seem somewhat high, but we should also consider that strategies that are predominantly perceptual might allow for greater depths to be searched without having such a major impact on the working memory. The straight-line strategy is probably less demanding of working memory than, for example, the travel time strategy which requires arithmetic to be done in the working memory while planning.

An often neglected, but important, aspect of problem solving are so-called "non-cognitive" aspects such as motivation (Funke et al., 2018). Motivation is understood as a threshold that regulates how much effort an agent is willing to expend to try to achieve the goal state of a task. In order to model the variation in motivation, each problem-solving agent received a threshold sampled from a negative binomial distribution with a mean parameter of 100, and a dispersion parameter of 1, which makes it positively skewed, reflecting the long tails that were observed in the empirical number of actions parameter. A summary of the fixed model parameters is given in Table 1.

There are multiple ways to switch between paths that induce variation in the action sequences that are not due to the planning strategy that is implemented but due to preference. Still, the noise due to differences in the method of switching between paths requires modeling. Four different

methods were observed to be used in the empirical data: Shifting paths using the reset button (reset); Shifting paths by backtracking the selected path to a node that allowed for the next path to be selected (backtrack); Shifting paths using the least number of actions needed (optimal); Shifting paths by first highlighting one path, then highlighting a second path, then removing the first path (memory). The strategy named backtrack was observed within the empirical data used in this study and also described in an example in the study by Chen et al. (2022). The last method, memory, was observed in the empirical data and was presumably used to offload working memory. The different ways of switching between paths could be viewed as being associated with the ”execution” part of problem solving. Examples of how the different methods would switch between the same paths are displayed in Figure 2. The visualizations show that the methods sometimes lead to differences in actions dependent on the method used to switch between paths. This figure also provides examples of the behavior of the model while problem-solving.

(a) reset

(b) backtrack

(c) optimal

(d) memory

Figure 2: Visualization of different methods of switching between paths. Action sequences are displayed row-wise. Cyan color: complete path, magenta color: incomplete path.

There could very well be differences in the proportions of each strategy used by the test-takers. The mixing proportions of strategies within the model were optimized by a combination of grid search and a random search process. Optimizing hyperparameters with random search

is rudimentary but has been shown to give better or equal results as grid search over the same domain within less computational time (Bergstra and Bengio, 2012). Due to constraints with respect to computational time resources, only 100 grid searches and 100 random searches were run. The mixture proportion that gave the most favorable fit indices with respect to a ranking of absolute deviation of the mean score, the mean number of actions, and the mean number of trials, was chosen as the best fitting mixture parameter. Grid search was done by selecting parameters from a fixed set of values. The grid search values was created by keeping proportions of random actions-strategy fixed at 0.05, while permuting elements in the following vectors: $[0.05, 0.20, 0.20, 0.25, 0.25]$, $[0.10, 0.20, 0.20, 0.20, 0.25]$, $[0.05, 0.20, 0.20, 0.20, 0.3]$, and $[0.10, 0.20, 0.20, 0.225, 0.225]$. Random actions were chosen to be fixed at 0.05 to limit the number of combinations to be tested, and since it seemed unlikely that this strategy should be more prevalent than any of the other strategies. However, for random search there was no such constraint, the random search process mixture component vectors were sampled with uniform probability with a $\text{Dirichlet}(\alpha = [1, 1, 1, 1, 1, 1])$. The results from three different mixture proportions are presented in Section 6.1.

5. Latent strategy inference

If process data are to be used to diagnose which strategy test-takers applied while problem-solving, we need some method to infer, or retrieve, which strategy generated the observed process data. The goal is to infer which type of latent program generated the data, or more specifically to infer which strategy the problem solving program was parameterized with. One way to approach this problem is to use a Multinomial naïve Bayesian classifier. Naïve Bayes is called "naïve" because it assumes that features are conditionally independent given the class. However, the classifier can be made less naïve by transforming the observed data into n-grams that respect the dependencies of adjacent symbols up to length n. N-grams are often used in text classification to account for sequential dependencies among words, and n-gram size has shown to have a positive effect on classification performance for natural language data (Peng et al., 2004). In the present study, we handle the process data as text, sequences of symbols, created by an alphabet that is made up of the actions that are possible to do in the Traffic task. In this case, the n-grams capture the context of actions regarding other actions. For further information on Multinomial naïve Bayes, see (Manning et al., 2008).

In brief, the strategy of using the classifier is as follows: simulate data on the problem-solving behavior for each strategy, train the naive Bayesian Classifier with n-gram processed data labeled with the strategy that generated the data, and apply the classifier to n-gram processed test-taker data to infer the latent strategy. A detailed description follows below.

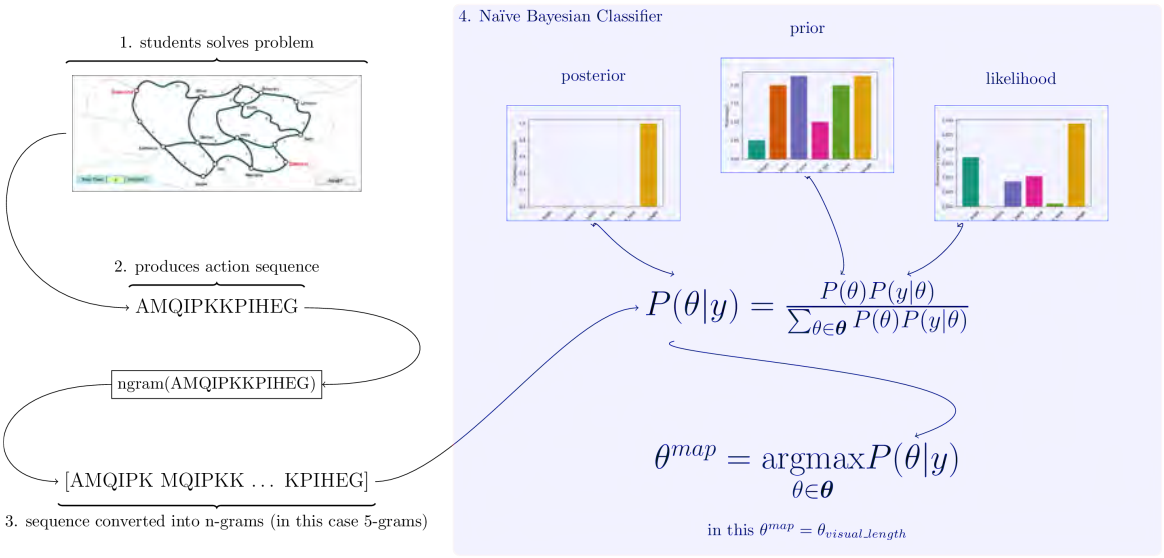


Figure 3: Illustration of the classification process.

In the present study, it is assumed that there exist a finite set of different strategies:

$$\boldsymbol{\theta} = \{\theta_{\text{random actions}}, \theta_{\text{random paths}}, \theta_{\text{travel time}}, \theta_{\text{straight line}}, \theta_{\text{angle}}, \theta_{\text{visual length}}\}$$

We seek to infer which strategy $\theta \in \boldsymbol{\theta}$ is most probable given that we have observed a sequence of actions y. We can calculate the probabilities of each strategy given the observed actions y by applying Bayes' theorem:

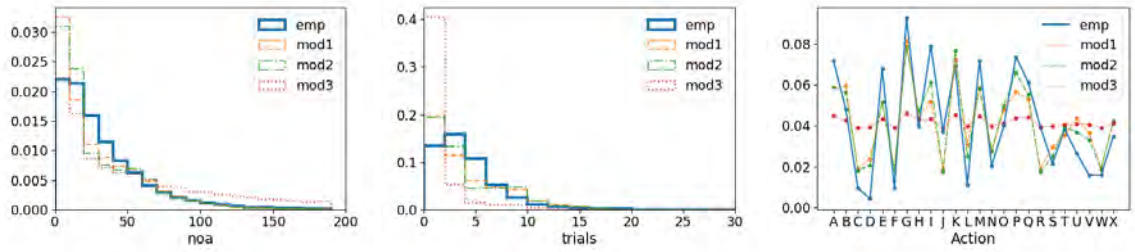
$$P(\text{strategy}|\text{actions}) = P(\theta|y) = \frac{P(\theta)P(y|\theta)}{P(y)} = \frac{P(\theta)P(y|\theta)}{\sum_{\theta \in \boldsymbol{\theta}} P(\theta)P(y|\theta)}$$

The prior $P(\theta)$ is in our case set from the best fitting mixture proportion parameter from the grid search and random search. The likelihood of actions (n-grams to be specific) given a strategy $P(y|\theta)$ was computed by maximum likelihood estimation from simulated data. Since the test data might contain features that were not observed in the training data, *Laplace smoothing* (Manning et al., 2008) with $\alpha = 1$ was used in the estimation of posteriors.

For an illustration of the classification process of individual test-takers data, see Figure 3. First (1) a problem solver attempts to solve the task and produces a sequence of actions that are (2) represented as a string of letters, and (3) converted into n-grams. The n-grams are then sent into Bayes' theorem, which computes the posterior probability of each strategy given the data, and finally, the strategy with the greatest posterior probability (MAP: Maximum a posteriori) is selected as the best estimate of which strategy that had generated the test-takers' data (4). In case more diagnostic information regarding the posterior probabilities of each of the strategies is of interest, these values can be retrieved from the step before classification.

6. Study 1: Simulations

Three simulation studies were conducted. The first simulation study, related to research aim 1, investigated whether the computational model provided a reasonably good account of the



(a) Number of actions (b) Number of trials (c) Action frequencies

Figure 4: Comparison of empirical (emp) versus simulated data (mod1, mod2, mod3). Plot (a) show a histogram of numbers of actions. Plot (b) shows a histogram over the number of trails. Max values for x-axes in plots (a) and (b) have been constrained to 200 and 30, respectively, hiding some outliers but increasing interpretability.

empirical data. The second simulation investigated the accuracy of inference of latent strategies (research aim 2). The third simulation study assessed the performance of the different strategies (research aim 3).

6.1. Simulation study 1: Model Evaluation

The aim of the first simulation study was to help set parameters of the model such that it produced behaviors that were similar to the behaviors produced by real test-takers. Stewart and West (2010) argues that a broad range of measures should be considered when evaluating computational cognitive models. In the present study, the measures selected for evaluation were task score (proportion solved tasks), the count of actions and count of trials (explored paths) before stopping, and frequencies of specific actions. To evaluate the computational model, 200 simulated data sets of size N = 24859 (equal to the sample size of real test-takers) were generated using the parameter settings presented in Table 1, and a mixture proportion that was given from the grid- or random search.

The goodness of fit indices used was the absolute error of differences of means between model-produced data and real test-taker data for task score, count of actions (NOA), and count of trials. Kullback-Leibler divergence D_{KL} quantifies the degree of dissimilarity between two probability distributions (Bishop, 2006) and was used as an additional measure of the similarity of the distribution of *action frequencies* between the empirical and simulated data. String metrics have previously been used to evaluate sequential process data (Hao et al., 2015), in the present study the sum of the pairwise edit distances between the empirical sequences and modeled sequences was used as an indicator of similarity, with Levenshtein distance as the measure of text distance. Furthermore the proportion of exact matching sequences between model generated data and data from real test-takers where calculated. Fit indices of three models only are included for brevity. Two of the best models and one of the worst models are presented in Table 2.

From Table 2, as well as Figure 4 we can see that both model 1 and model 2 show similar behavior and a better fit with respect to the empirical data compared to model 3. The fit is however not perfect as indicated by the histograms. The counts of actions and counts of trials

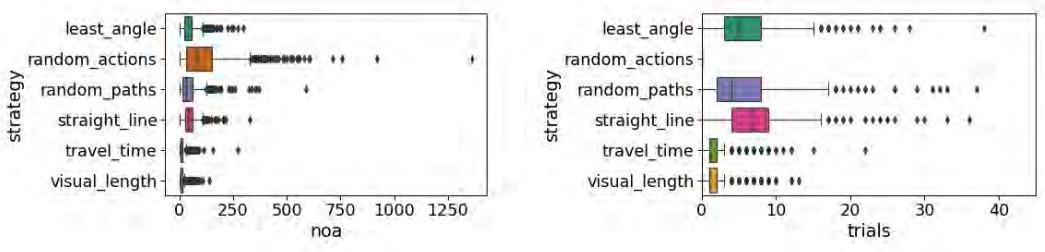
Table 2: Fit indices of three models

Model	Mixture proportions	Score	NOA	Trials	D_{KL}	Match	Lev
82	[.05, .2, .225, .1, .2, .225]	.001	.305	.004	.074	.193	2.563
114	[.072, .094, .036, .22, .123, .455]	.000	.076	.106	.060	.190	2.591
100	[.661, .018, .04, .028, .051, .201]	.435	47.776	2.668	.173	.210	5.108

Score, NOA (number of actions), Trials show absolute differences of means to empirical data;
D_{KL}: Kullback-Leibler divergence of action frequencies from empirical frequencies;
Match: proportion exact matches; Lev: Levenshtein Distance $\times 10^{-10}$.
Leading zeros were removed to reduce the width of the table.

are more right-skewed in models 1 and 2 compared to the empirical distribution, though better fitting compared to model 3. The frequencies of actions from models 1 and 2 show a more similar pattern to the empirical data than action frequencies from model 3, but there is not a perfect overlap.

To provide further information on the similarity between the empirical and model-generated distributions further analyses were run on model 1 and model 2. Modeling the task score data with a beta-Bernoulli model and estimating the difference of the probability of success-parameter between modeled and empirical data gave results that a zero difference was within a 95% Credible Interval (CI) for both model 1 and model 2. Modeling numbers of actions and number of trials with a negative binomial and comparing the probability of a zero difference between parameters from empirical and model-generated data, gave results that for numbers of actions the equality of mean parameters was within a 95 % CI for both model 1 and model 2, however, for the modeling of the count of trials a zero difference in the mean parameter was within 95% CI only for model 1. The equality of dispersion parameters between empirical and model-generated data was however outside a 95% CI for both model 1 and model 2. Since model 1 produced more similar counts of trials than model 2 compared to the empirical data, model 1 was chosen as the best-fitting model and used in subsequent analyses.



(a) Number of actions

(b) Number of trials

Figure 5: Number of actions and number of trials for each strategy. Results from model 1

Figure 5 present boxplots that show how each of the strategies contributes to the number of actions, and the number of trials. Results are from Model 1. The plots show the travel time- and visual length strategy was associated with lower counts of actions and counts of trials, compared to the other strategies.

The overall impression of model 1 and model 2 is that they produce data that, in many ways, are similar to the observed data, and which for the present study is considered to be satisfactory.

6.2. Simulation study 2: Strategy Retrieval

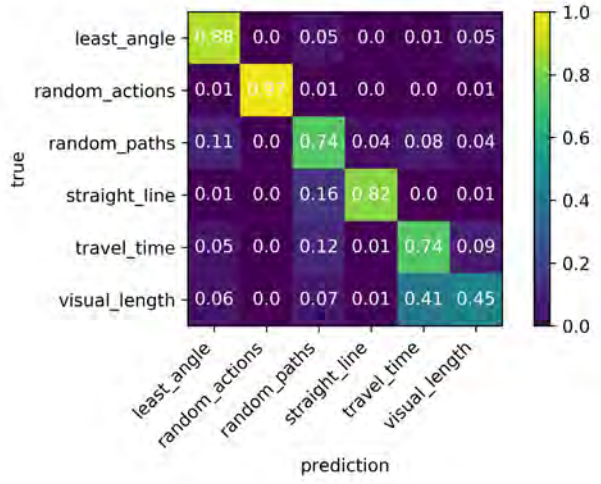
6.2.1. Original version of the Traffic task

To investigate the accuracy of retrieval of latent programs (strategies), data were simulated using parameter values from Table 1, and then classified with the Multinomial naïve Bayesian classifier augmented with n-grams, as described in Section 5. Here n-grams of size n refer to computing all n-grams from 1 to n, for example an n-gram with $n = 3$ includes all combinations of 1-grams, 2-grams and 3-grams.

A total of 10 simulated data sets of sample size $N = 24859$ were created. Sets 1 to 9 were used to "train" the classifier by calculating the likelihoods of the n-grams given each strategy class. The classification accuracy (% correctly classified) was then assessed by classifying the data from the 10th set. The classification accuracy was evaluated for n-grams 1:1, 1:2, ..., 1:15. The results are presented in Figure 6. The classification accuracy for n-gram sizes yielded an accuracy of 0.52 for 1-grams and increased with greater n-grams to an accuracy of 0.72 for $n = 15$.

n	acc	n	acc	n	acc
1	0.52	6	0.68	11	0.71
2	0.62	7	0.70	12	0.71
3	0.65	8	0.70	13	0.72
4	0.66	9	0.71	14	0.72
5	0.67	10	0.71	15	0.72

(a)



(b)

Figure 6: Panel (a): classification accuracy (acc) for different n-grams (n). Panel (b): confusion matrix when preprocessing the data with 1:15-grams, the matrix is normalized over rows.

Results presented in Table (a) in Figure 6 show that it is important to take the sequential dependencies into account to improve the classification of problem-solving strategies. The confusion matrix provides a more detailed account of the classification accuracy. We can for example see that the random actions strategy and least angle strategy are predicted correctly with an accuracy of 97%, and 88%, respectively. The prediction of the strategy travel time is 5% of times wrongly predicted as least angle, 12% predicted as random paths, 1% wrongly predicted as the straight line strategy, and 9% predicted as visual length strategy. We can also see that the visual length strategy is quite often (41%) wrongly predicted as the travel time strategy.

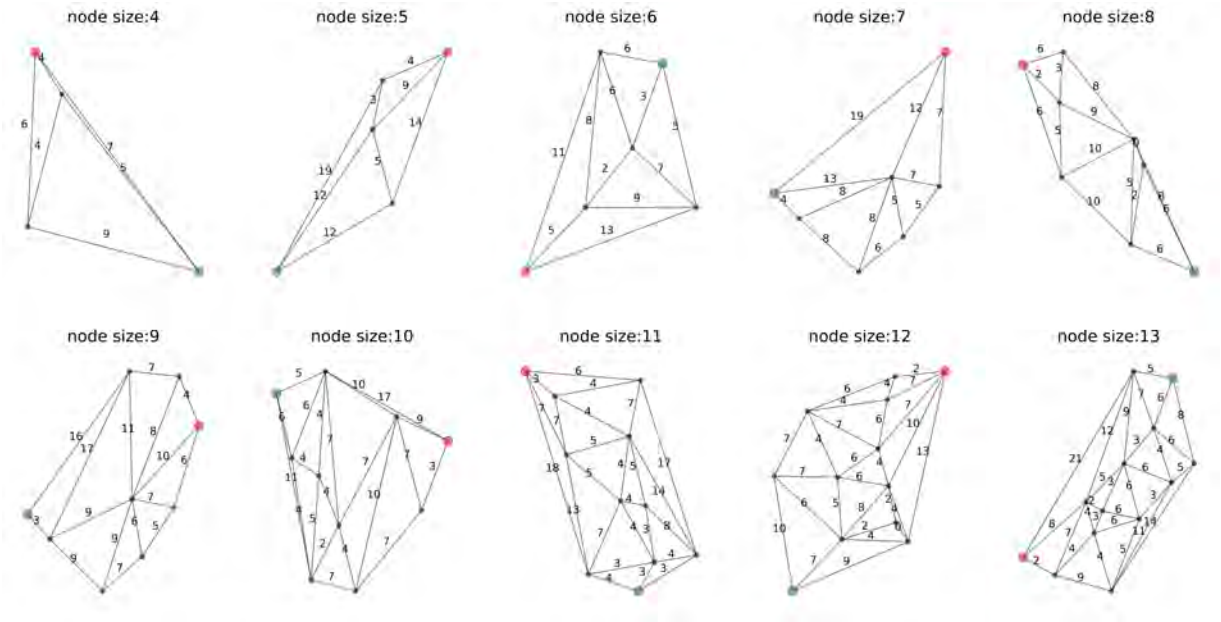
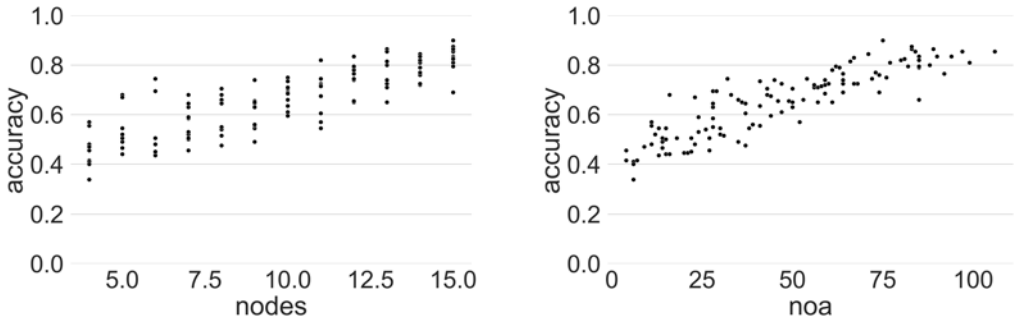


Figure 7: Examples of generalized versions of the Traffic task. The green node indicates the start and the red node indicates the goal. Numeric labels indicate the travel times.

6.2.2. Generalized version of the Traffic task

It is not only important to understand the classification performance on the specific Traffic task that appeared in the PISA assessment, but also how it might be affected if the items had other properties. To investigate how the effects of changes in item design relate to classification accuracy, generalized versions of the Traffic task were generated by changing the node size and adding variation to the layout of roads and travel times, see Figure 7 for examples. For an explanation of the procedure used to generate generalized versions of the tasks, see Algorithm 3 in Appendix 9.2.

Traffic tasks with node size varying from $n = 3$ to $n = 15$ were generated, 10 of each size. 1,000 problem solvers were then simulated to try to solve each of the generalized tasks. Plots were drawn to assess how classification accuracy was affected by both problem size and the number of actions carried out by the problem solvers. The results are presented in Figure 8, where panel (a) shows a positive relationship between the classification accuracy and node size of the problems. In panel (a) we can also note that there is also variation in classification accuracy between problems of the same size. In panel (b) a similar positive trend can be observed between accuracy and the mean number of actions done by problem solvers, suggesting that classification improves when more actions are produced.

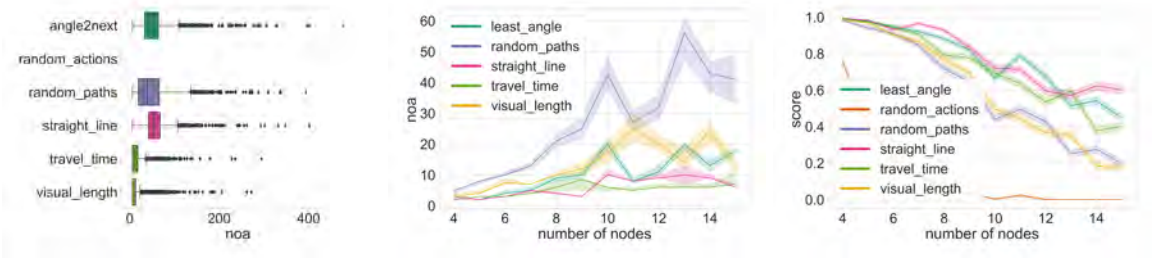


(a) (b)

Figure 8: Classification accuracy versus the number of nodes in the problem (a). Classification accuracy versus the *mean* number of actions (noa) received by the problem (b).

6.3. Simulation study 3: Strategy Performance

To assess the performance of each of the strategies, the number of actions and the proportion of solved items were used as performance measures. The simulated data were the same as those used in the previous Section 6.2.



(a) (b) (c)

Figure 9: (a) Number of actions (noa) for each strategy before solving the PISA Traffic task. (b) Number of actions for each strategy on generalized problems as node size increases (random actions strategy was removed from this plot to increase interpretability). (c) Mean score for each strategy on generalized problems as node size increases. The intervals in panel (b) and (c) show 95% confidence intervals

Figure 9 panel (a) shows the number of actions for each strategy that solved the original PISA Traffic task. The strategies that had the best performance on the original PISA problem-solving task in terms of the proportions of solved tasks were the travel time strategy and the visual length strategy, median = 0.88, and 0.89, respectively. Regarding the efficiency of problem-solving indicated by the median number of actions before solving a task, travel time was the most efficient in the original Traffic task (median = 6). The random actions strategy was the

lowest-performing strategy, only 1 out 12431 simulated problem solvers managed to solve the original Traffic task.

On generalized versions of the task, the best scoring strategy was the straight-line strategy with 0.79 proportion of solved tasks. The most efficient strategies in the generalized tasks were the straight-line strategy with a median of 7 actions before solving a task.

Figure 9 panel (b) shows the number of actions for each strategy on generalized tasks as the node size of the problem increases. We can see that both the travel time strategy and the straight-line strategy are less affected by the increased size of the problem. The random actions strategy has been removed from panel (b) to increase the interpretability of plots. Panel (c) shows how the mean score in the tasks is related to increasing the node size of the problems. Table 3 provides complementary summary statistics to Figure 9.

Table 3: Proportion of solved items for simulated data on original (score og.) and generated tasks (score gen). Median number of actions for solved tasks for original PISA Traffic task (noa og.) and generalized (noa gen.) tasks.

strategy	score og.	score gen.	noa og.	noa gen.
least_angle	0.61	0.76	49	7
random_actions	0.00	0.14	8	26
random_paths	0.63	0.59	39	13
straight_line	0.59	0.79	54	5
travel_time	0.88	0.72	6	4
visual_length	0.89	0.60	12	8

7. Study 2: Applying latent strategy inference to PISA data

To apply latent strategy inference to the real test-takers' data, and to collect validity evidence regarding classifications, it was investigated how the inferred strategies of real test-takers' behavior related to their task performance and overall PISA problem-solving performance. Test-takers' number of actions and number of trials are presented in boxplots and can be compared with the boxplots in Figure 12. Boxplots were also drawn for response times since it could be interesting to see how the real test-takers' response times are distributed for test-takers with the same strategy classification. The data cleaning process for the empirical data is described in Appendix 9.1.

7.1. Latent strategy inference on real test-takers

The strategy (most probably) used by real test-takers was inferred by the classification procedure outlined in Section 5, with the exception that all 10 sets of simulations were used to train the Bayesian classifier before using it on the data from real test-takers. The results from the classification proportions are shown in Table 4.

Table 4: Proportions of classified strategies from real test-takers

predicted_strategy	proportion
least_angle	0.29
random_actions	0.01
random_paths	0.46
straight_line	0.08
travel_time	0.08
visual_length	0.07

7.2. TASK PERFORMANCE PREDICTED BY STRATEGY

To investigate whether the test-takers differed in the probability of solving the Traffic task depending on their predicted strategy, a beta-binomial model (Levy and Mislevy, 2017) was used to model the probability of solving the Traffic task for test-takers that had the same strategy classification. In this case, the success parameter p governs the probability of answering the item correctly. The following model was used:

$$y_k \sim \text{Binomial}(N_k, p_k)$$
$$p_k \sim \text{Beta}(1, 1), \text{ for } k = 1, 2, \ldots, 6$$

A uniform beta distribution was used as prior for the success parameter p for each strategy k. N_k indicates the number of test-takers classified as using strategy k. The results show that the strategy classification of the test-takers is related to differences in the probability of solving the Traffic task. The parameters for the Beta-Binomial model were estimated using grid approximation.

The random action strategy had a low probability of success $\hat{p} \approx 0.11$. To put this number into context, the probability of success of all test-takers was 0.71 which is about 6.5 times higher than the estimate of the probability of success of test-takers classified as using the random actions strategy. Test-takers classified as using the travel time strategy had the highest probabilities of solving the task, $\hat{p} \approx 1$, that is, everyone that was classified as using travel time solved the task. For the success parameters of the remaining strategies see Figure 10

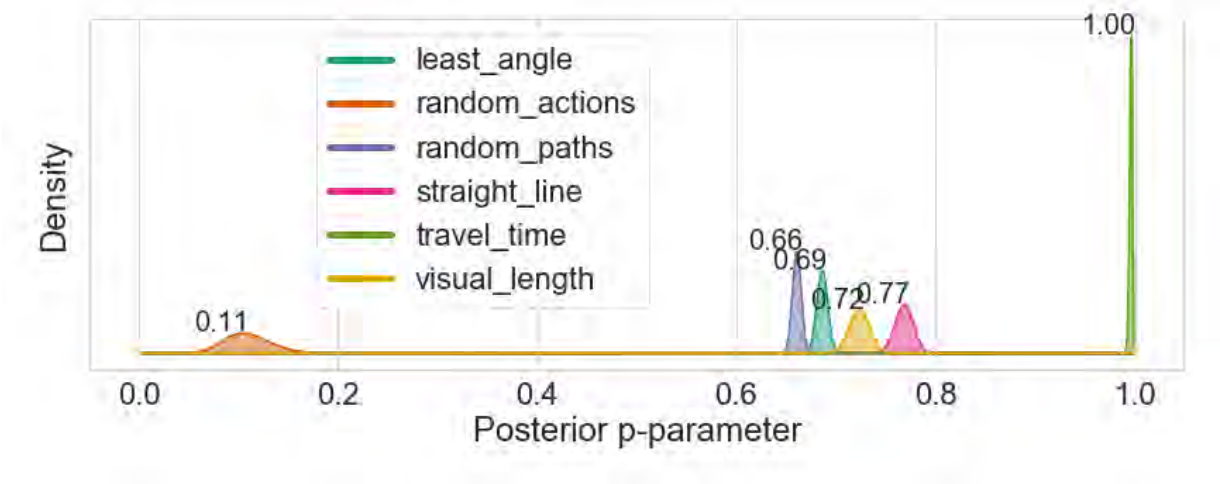


Figure 10: The posterior density of the success parameter p for each of the strategies. The figures at the top of the distributions indicate the maximum a posteriori of the success parameter $\hat{p}$.

7.3. PISA PERFORMANCE PREDICTED BY STRATEGY

To investigate the effect of strategy on PISA performance the following regression model was used:

$$
\begin{aligned}
&pisa\_score_i \sim \text{Normal}(\mu_i, \sigma) \\
&\mu_i = \alpha + \beta_{strategy[i]} + \beta_{task\_score} \times task\_score_i \\
&\alpha \sim \text{Normal}(500, 100) \\
&\beta_{strategy} \sim \text{Normal}(0, 100) \\
&\beta_{task\_score} \sim \text{Normal}(0, 100) \\
&\sigma \sim \text{Half-Normal}(100)
\end{aligned}
$$

In this model, the subscript i indexes test-takers $1, 2 \ldots 24859$. The $pisa\_score_i$ refers to the PISA score of the ith test taker, which is modeled as normally distributed with a mean that depends on the intercept α, the strategy classification of the ith test-taker $\beta_{strategy[i]}$, and the task score of the ith test-taker $task\_score_i$. The coefficient $\beta_{strategy}$ represents the effect the specific strategy classification of a test-taker has on their PISA score. The coefficient β_{task_score} is used to account for the effect a correct score on the Traffic has on the PISA score. Since PISA scores are known to be normally distributed around a mean of 500 with a standard deviation of 100, priors on the intercept α and the parameter for the standard deviation σ were set to reflect this knowledge. Since we do not have any prior knowledge regarding the potential effects of strategy on the PISA score, we chose a Normal(0,100), for both β-coefficients, which reflects a prior belief that these effects are centered around 0 but allows for large variations around the mean.

The task score was included as a predictor since performance in the Traffic task is influencing the estimate of the PISA performance scores. We are interested in the effect of the strategy independent of whether the item was solved or not. A model without task score as a predictor was however also fitted and showed the same pattern for the $\beta_{strategy}$-coefficients, but with

slightly larger effects of each strategy, and an especially larger negative effect from random actions. Results are presented from the model that includes task score as a predictor.

The model was implemented in Stan (Carpenter et al., 2017) which uses Hamiltonian Markov Chain Monte Carlo to estimate the posteriors.

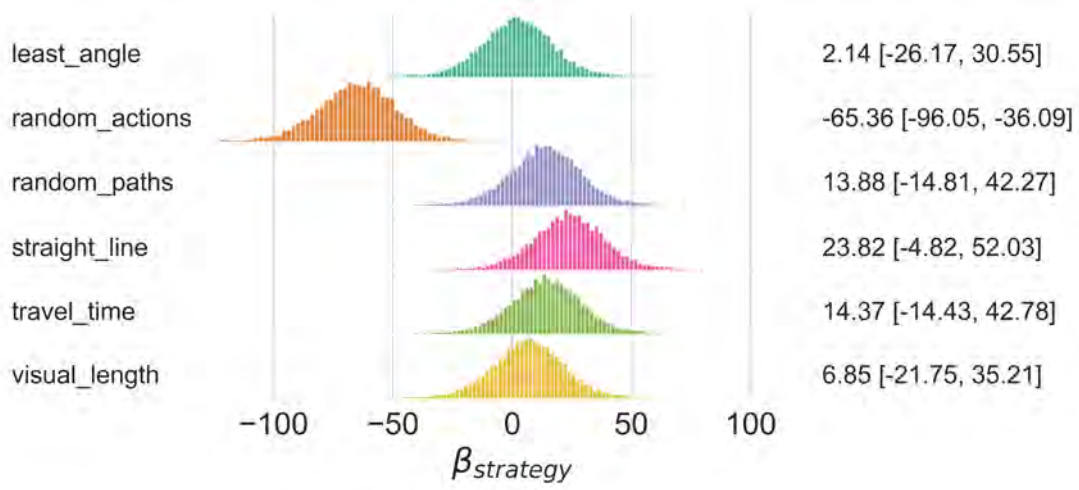


Figure 11: The effects of strategy on the PISA score, posterior deviations from the global intercept. On the right-hand side point estimates are provided (posterior mean with 95% CI in brackets).

The results from predicting the PISA performance from the problem-solving strategy are presented in Figure 11. Histograms of the posterior β parameter for each of the strategies are displayed together with mean and 95% credible Intervals (CI). The only strategy with a β parameter that differed from 0 with probability > 0.95 was the random actions strategy with $\beta_{random\_actions} : \mu = -65.36,\ CI_{95\%} = [-96.05, -36.09]$. The global intercept was estimated to $410.10,\ CI_{95\%} = [373.12, 448.12]$ effect of task score was estimated to $98.81,\ CI_{95\%} = [96.32, 101.32]$.

7.4. Number of actions, numbers of paths, and response time

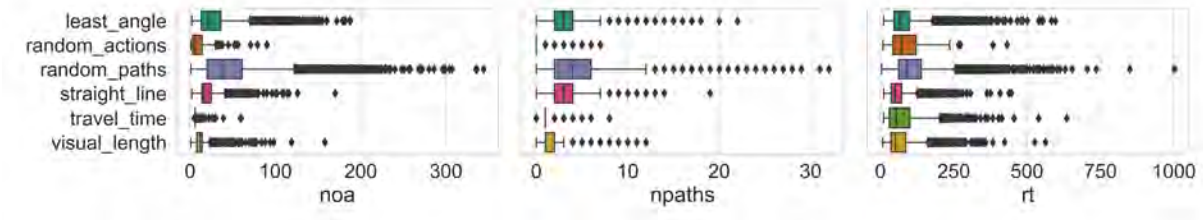


Figure 12: The number of actions (noa), number of trials (npaths), and the response time (rt) - the total time spent on the task for test-takers sharing the same strategy classification.

If we look at Figure 12, and Table 5, we see that the number of actions of test-takers classified as using a random actions strategy has a 0 median number of trials, indicating that most of

the test-takers classified as using this strategy did not complete any paths between Diamond and Einstein. It should also be noted that while the straight line strategy has a greater median number of actions and greater median number of trials, the median response time of the straight line strategy is shorter compared to the response time of the travel time strategy.

Table 5: Number of actions (noa), Median number of trials, median response time (rt) for the predicted strategies.

predicted_strategy	noa	trials	rt
least_angle	22.00	3.00	66.80
random_actions	6.00	0.00	71.30
random_paths	38.00	4.00	91.00
straight_line	16.00	3.00	49.70
travel_time	6.00	1.00	54.00
visual_length	12.00	2.00	51.40

8. Discussion

The present study aimed to 1) create a theoretically informed computational model that simulated different problem-solving strategies in a PISA 2012 problem-solving task; 2) investigate whether it was possible to retrieve the strategy that was underlying the observed problem-solving behaviors, and whether the performance of the strategy retrieval was affected by changing item design; and 3) evaluate how different problem-solving strategies performed in both the original Traffic task and in generalized versions. Finally, the study aimed to 4) apply the model to data from real test-takers to infer which strategy had the highest probability of having generated the test-takers' actions, and whether the test-takers' inferred strategy was related to performance in the Traffic task and general PISA problem-solving performance.

8.1. Discussion of Study 1: Simulations

In the first simulation study, the behavior of the computational model was evaluated by comparing it to the behavior of real test-takers. It was found that all of the strategies except random actions were sufficient to solve the task (random action can theoretically be able to solve the task, but the probability is very low). Overall, the computational model seemed to simulate data that were similar to the empirical data according to the statistics and plots that were considered. Equality between simulated data and test-taker data regarding the probability of success, mean numbers of actions, and numbers of trials, was within a 95% CI. The percentage of simulated data that exactly matched the empirical problem-solving data was around 20%, which might seem like a low figure. However, it cannot be expected that the exact match statistic would be very high since the complexities of real human behavior, in combination with the many potential ways of creating sequences, allow for many unique sequences to be created. In the data from real test-takers, around 70% of the sequences were unique. We should also note that the proportion of exact matches is partially limited by how many simulations were run, which were limited due to time constraints. Since not all parameters of the model were optimized it is possible to

achieve a better fit; however, since this comes at a high computational cost with the current parameter estimation method it was not practical to pursuit a better fit.

The simulations that investigated the retrieval accuracy of the Naïve Bayes classifier showed that the accuracy improved with increased n-gram size. This demonstrates that it is important to take sequential dependencies into account when making inferences from this kind of process data. That the classification accuracy was not higher than 72% can be explained by identifiability issues. For example, the random paths strategy can produce any of the paths that the other strategies are capable of producing (except random actions). For example, the random paths strategy might be fortunate and find the correct answer on the first attempt the same way the travel time strategy often does. This could also be true for some of the other strategies. For example, we can see from the confusion matrix that for 41% of the classifications of the visual length strategy was falsely predicted to be the travel time strategy. This means that the retrieval accuracy of all strategies, using this kind of data and this specific method of inference, will not be perfect. However, for most strategies, the classification accuracy was between 70% and 97%, showing that there are differences regarding the extent to which inferences can be trusted. When interpreting the scatter plots in Figure 8, it can be seen that there is a positive linear trend between the number of nodes and classification accuracy, and also a linear trend between the mean number of actions on problem and classification accuracy. This seems reasonable since an increased node size of problems leads to a problem with an increased number of connections between nodes, and more paths to search among. For bigger problems, more actions have to be made before a solution is found or motivation runs out, which provides more opportunities for deviations in behavior that can be used to identify the strategies. Consider a test-taker who only completed two actions, it would be impossible to determine which strategy the test-taker used with any degree of certainty since most strategies would start similarly. However, when many actions are completed, the action sequences are more likely to diverge due to differences in action selection between strategies. The results from the classification accuracy on generalized problems show a variation in classification accuracy for different problems, sizes of problems, and the number of actions that were needed to solve the problem. The problem with the highest accuracy had an accuracy of 90%; this specific problem had 15 nodes, 37 edges, and 25705600 unique paths between the start and goal node. This result shows that it is possible to create problems with an increased diagnostic capability to identify solution strategies.

The results from the simulation study that investigated the performance of strategies showed that, in the original Traffic task, all strategies, except random actions, were sufficient to solve the task. This means that the test-takers could solve the task using at least five different strategies. Interestingly, the random paths strategy does not use any information when selecting among paths, which shows that it is possible to solve the task using a basic strategy of random trial and error between different paths; the only thing that is needed to solve the task can be a little bit of luck or a lot of effort. As can be seen in panel (a) in Figure 9, on the PISA Traffic task the random paths strategy is actually on average *better* than a heuristic straight-line strategy. An explanation is due to the specific design of the PISA Traffic task. Assuming the search starts from the node Diamond, then the straight line strategy is biased toward first trying out paths that start with edge B. On the other hand, the random paths strategy has a 0.5 probability of selecting edge A. Since edge A is a sub-path of the solution path, and there are not many paths to choose from there on, the random paths strategy will, on average, find the solution quicker. The result showing that the travel time strategy had a good overall performance is less surprising given that this strategy optimizes what is important for solving the task, namely, choosing the path

with the lowest travel time cost. The visual length strategy was effective in solving the original Traffic task but had a lower probability of solving the generalized versions. Other strategies such as straight line and least angle strategy seem to be on par or better with the travel time strategy on generalized problems. The results show that different strategies can be both efficient and effective. From the plotting graphs showing how the node size of the problem relates to the proportion of solved tasks, we find that the difficulty of problems increases as the node size increases, and that this applies to all strategies. However, some strategies, such as random paths and visual length have a harder time solving the more complex tasks. If it would be of interest to make the task more difficult, so that it is not as likely to be solved by a simple strategy (random paths), a recommendation would be to increase the node size of the task, or the number of connections between nodes.

8.2. Discussion of Study 2: Application

The second study classified data from real test-takers and investigated the effect of classification on the probability of solving the task, and the effect of classification on overall test performance. Most test-takers were classified as using the random paths and least angle strategy. Comparatively few test-takers were classified as using the visual length strategy, and very few test-takers were classified as using the random actions strategy.

That the test-takers classified as implementing random actions had a very low probability of solving the task seems logical with respect to the definition of the strategy and the results from the simulations, and provides further validity evidence regarding the classifications. However, it is somewhat surprising that the probability of correct was as high as 0.11 which indicates that some test-takers classified as random actions actually solved the task. This suggests that the random actions strategy can be viewed as a residual class which includes some test-takers that had behavior that was very idiosyncratic compared to any other strategy, and perhaps had a fragmented way of building their paths which would be similar to the behavior generated by a random actions strategy.

Further validity indications are given in Figure 12 which shows that test-takers classified as using the random paths strategy used a relatively greater number of actions, and the test-takers using the travel time strategy used a relatively fewer number of actions, both results are in line with the results from the simulations. However, we can see that the test-takers classified as using random actions had quite a low number of actions, and 0 median number of trial attempts, which could indicate low motivation, misunderstanding, or behaviors driven by other goals than trying to solve the task. The fact that the response time of random actions is similar to other classifications could indicate increased deliberation time or idleness between actions. We can also note that the median response time of test-takers classified as using the travel time strategy is longer compared to both the straight line- and visual length strategy even though the travel time strategy had a lower median number of actions, this result also indicates an increased deliberation time form test takers using the travel time strategy which seems reasonable given that the travel time strategy should be more cognitively demanding.

Regarding the effect of strategy use on PISA performance, the random actions strategy was the only strategy with a strong deviation from the mean and had a negative effect on the PISA score. This means that test-takers behaving as if they were using the random actions strategy in the Traffic task had a lower overall problem-solving test performance. The result seems reasonable, given that the random actions could indicate unmotivated behavior or misconceptions

about the task. The fact that the other strategies did not affect the overall PISA problem solving score indicates that the different strategies used in the Traffic task were not strongly related to overall problem-solving skills. A reasonable explanation in our case would be that using an efficient strategy for one specific kind of problem might not have much in common with the ability to solve other kinds of problems. This result is in line with Ulitzsch et al. (2021), wherein the test-takers' strategies derived from performed actions were not predictive of their proficiency level.

Overall, we find that using simulations from a computational cognitive model makes it possible to find interesting and useful information about the possibility to identify and evaluate problem-solving strategies. The development of a cognitive model also leads to a greater appreciation of the complexity of the data-generating process and the kind of thinking processes that might govern the observed problem-solving behaviors.

8.3. Limitations and Future Research

A limitation of the present study is that the selection of strategies is non-exhaustive and that there could exist other strategies that were not considered. This has the effect that all test-takers that used some other strategy that is not represented among the a priori defined strategies will be forcedly classified into one of these strategies. Thus, removing or adding strategies would affect the results of this study. The limitation due to the non-exhaustive consideration of all strategies is related to another limitation, which is that no empirical information was retrieved from the real test-takers subjective experience regarding how they would account for their problem-solving strategies. Ideally, think-aloud protocols (Ericsson and Simon, 1984) in conjunction with behavioral observations of the test-takers could be used to improve and give further credibility to the selection of strategies to be included in the model. Eye-tracking data would also provide valuable and detailed information regarding the test-takers' search strategies. Thus, future studies could improve upon the current methods by collecting additional empirical data from real test takers which complement the behavioral data derived from the existing click-stream data that was used in this study.

For reasons expressed in the introduction, no standard path planning AI algorithm was used in this study to compare human search behavior against the behavior of computer algorithms. However, future studies could make further investigations with respect to how well standard algorithms from computer science can account for human search behavior in the Traffic task. Less memory-intensive versions of A*, weighted A*, and iterative deepening A*, could be interesting alternatives.

A limitation of the current study is that time latencies between actions are not considered. Ulitzsch et al. (2021) have shown that within-task response times are important for differentiating between response processes. This is also indicated by the planner clusters identified by Lundgren and Eklöf (2020). However, there are conflicting results with respect to how important time is in classification, as both Salles et al. (2020) and Qiao and Jiao (2018) found time-related features to be unimportant in classification. When modeling response times, attempts could be made to model the timing intervals between actions using cognitive architectures which provide theoretically sound estimates of latencies between actions and cognitions, and then investigate whether this more refined modeling leads to improved inferences. Taking response latencies into consideration would also be a move towards "strong equivalence" between models and the real thing, a stronger argument that models over cognition are valid (Pylyshyn, 1984).

An improved method of parameter fitting of the computational model could have provided a better fit for the model developed in this study. There are methods, such as approximate Bayesian computation (Kangasrääsiö et al., 2019), or genetic algorithms, that can be used to estimate or optimize parameters in models without explicit likelihood functions. Future studies could explore if such methods are practical to use for models similar to the one developed in this study. Many parameters that could be of potential diagnostic interest, such as motivation, probability of forgetting, and indecisiveness, were set as fixed values and not estimated at the individual level, perhaps with more data and improved estimation techniques, this could be possible.

Apart from the clear opportunities to use computational cognitive models to retrieve a deeper and more detailed understanding of test-takers' skills and approaches to problem solving, there are also great opportunities for item development, e.g. by creating items that are optimally informative about latent constructs, for a formal approach to designing optimal tasks see Rafferty et al. (2014).

There are practical concerns about using models that address process data on the level of task performance. Creating computational models is a relatively time-consuming task, and there is often limited generalizability of task-level models to other kinds of tasks. See Leighton and Gierl (2007) for a more extensive discussion on this subject. However, task-level models still play an important role in the validation of score meaning (Kane and Mislevy, 2017). Without creating models that can account for and explain the link between observed process data and latent constructs, there are few convincing arguments that support that a test item within an assessment actually does measure the construct they are supposed to measure.

9. Appendices

9.1. Data cleaning and pre-processing of empirical data

The PISA data file contains a time-stamped log of the clicks that were made, as well as a boolean vector representing the state of highlighted edges in the graph. For a more detailed explanation of the data and how it relates to the task, see (Lundgren and Eklöf, 2020). The log file contained information about clicks on areas of the Traffic task that does not result in any change in the task environment; non-effective clicks outside the edges, or outside the reset button, for example, clicks on text, or the background image, help menu, etc. All of these non-effective actions were removed as they were not of interest when modeling the problem solving strategies. The actions that were retained were clicks on the reset button and clicks on path segments (edges). When exploring the data, it was found that some of the test-takers double-checked the correct answer. These double checks were removed since they fell outside the conceptualization of the problem-solving strategy. Some of the test-takers started out by clicking the reset button multiple times, which seemed unlikely to be driven by their strategy to find a path, and thus any of these initial reset actions were removed.

In the present study, the main interest was in using sequences of events. Thus, specific time observations were discarded. This resulted in data in the form of an ordered list of actions. The list of actions was represented as a string by coding each action type with a single symbol and aligning them in their temporal order, the same type of representation as the "action string" used by Hao et al. (2015). In the present study, actions were coded with the letters $\{A, B, \ldots, X\}$, X refers to clicking the reset button, and the other letters refer to clicking on specific edges.

For the mapping between letters and actions, see the annotations in Figure 1. As an example, (assuming no previous actions were taken), a sequence of actions leading up to the highlighted path in Figure 1 is represented by `BTIPK`.

Data were retrieved from the official PISA website: https://www.oecd.org/pisa/pisaproducts/database-cbapisa2012.htm. All data that passed through the above described pre-processing were used, which resulted in N = 24,859 observations.

9.2. Algorithm for creating generalized versions of Traffic task

Algorithm 3 Generate a Traffic task

1: chose the desired node size of problem
2: angle $\sim U(0, 2\pi)$ ▷ sample direction to walk
3: radius $\sim U(3, 10)$ ▷ sample length of walk
4: $x \leftarrow$ radius $\times$ cos(angle), $y \leftarrow$ radius $\times$ sin(angle) ▷ convert to Cartesian coordinate
5: position $\leftarrow [x, y]$
6: nodes.append(position) ▷ save node position
7: **while** node size $<$ desired node size **do**
8: repeat step 2 to 6 if it does not lead to positioning nodes too close to each other
9: **end while**
10: create a graph from list of nodes by using Delaunay triangulation.
11: $\text{edge}_{travel\_time} = edge\_length + noise \sim U(\{-2, -1, 0, 1, 2\})$ ▷ give each edge a travel time

10. Supporting materials

See the repository at https://doi.org/10.5281/zenodo.6670627 named *Supporting materials for "Latent program modeling: Inferring latent problem-solving strategies from a PISA problem-solving task"*, Digital Object Identifier (DOI): 10.5281/zenodo.6670627.

11. Acknowledgments

I would like to thank my supervisors Hanna Eklöf and Inga Laukaityte for help with proofreading and giving feedback on the organization of the text in this article.

References

American Educational Research Association, t. A. P. A. and the National Council on Measurement in Education. 2014. *Standards for educational and psychological testing*. American Educational Research Association.

Baker, R. 2010. Data mining. In *International Encyclopedia of Education (Third Edition)*, Third Edition ed., P. Peterson, E. Baker, and B. McGaw, Eds. Elsevier, Oxford, 112–118.

Baker, R. and Siemens, G. 2014. *Educational Data Mining and Learning Analytics*, 2 ed. Cambridge Handbooks in Psychology. Cambridge University Press, 253–272.

Basso, D., Bisiacchi, P. S., Cotelli, M., and Farinello, C. 2001. Planning times during traveling salesman's problem: Differences between closed head injury and normal subjects. *Brain and Cognition 46,* 1-2, 38–42.

BERGSTRA, J. AND BENGIO, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13,* 10, 281–305.

BISHOP, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer.

CARPENTER, B., GELMAN, A., HOFFMAN, M. D., LEE, D., GOODRICH, B., BETANCOURT, M., BRUBAKER, M., GUO, J., LI, P., AND RIDDELL, A. 2017. Stan: A probabilistic programming language. *Journal of Statistical Software 76,* 1, 1–32.

CHEN, Y., ZHANG, J., YANG, Y., AND LEE, Y.-S. 2022. Latent space model for process data. *Journal of Educational Measurement*. Advance online publication. doi:10.1111/jedm.12337.

COWAN, N. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences 24,* 1, 87–114.

DALTON, R. C. 2003. The secret is to follow your nose: Route path selection and angularity. *Environment and Behavior 35,* 1, 107–131.

DE GROOT, A. D. 1978. *Thought and Choice in Chess*, second ed. Mouton Publishers.

DUBEY, R. K., SOHN, S. S., THRASH, T., HOLSCHER, C., KAPADIA, M., AND BORRMANN, A. 2022. Cognitive path planning with spatial memory distortion. *IEEE Transactions on Visualization and Computer Graphics*. Advance online publication. doi:10.1109/TVCG.2022.3163794.

DYE, H. A. 2007. A diagrammatic reasoning: Route planning on maps with act-r. In *Eighth International Conference on Cognitive Modeling*, R. L. Lewis, T. A. Polk, and J. E. Laird, Eds. 217–218.

EDELKAMP, S. AND SCHRODL, S. 2011. *Heuristic search: theory and applications*. Elsevier.

ERCIKAN, K. AND PELLEGRINO, J. W. 2017. *Validation of score meaning for the next generation of assessments: The use of response processes*. Taylor & Francis.

ERICSSON, K. A. AND SIMON, H. A. 1984. *Protocol analysis: Verbal reports as data.* MIT Press.

FANG, G. AND YING, Z. 2020. Latent theme dictionary model for finding co-occurrent patterns in process data. *Psychometrika 85,* 3, 775–811.

FUNKE, J., FISCHER, A., AND HOLT, D. V. 2018. Competencies for complexity: problem solving in the twenty-first century. In *Assessment and teaching of 21st century skills*. Springer, 41–53.

GÄRLING, T. 1989. The role of cognitive maps in spatial decisions. *Journal of Environmental Psychology 9,* 4, 269–278.

GREIFF, S., WÜSTENBERG, S., AND AVVISATI, F. 2015. Computer-generated log-file analyses as a window into students' minds? a showcase study based on the pisa 2012 assessment of problem solving. *Computers & Education 91*, 92–105.

HAO, J., SHU, Z., AND VON DAVIER, A. 2015. Analyzing process data from game/scenario-based tasks: an edit distance approach. *Journal of Educational Data Mining 7,* 1, 33–50.

HE, Q., BORGONOVI, F., AND PACCAGNELLA, M. 2021. Leveraging process data to assess adults' problem-solving skills: Using sequence mining to identify behavioral patterns across digital tasks. *Computers & Education 166*, Article 104170.

HOCHMAIR, H. AND FRANK, A. U. 2000. Influence of estimation errors on wayfinding-decisions in unknown street networks–analyzing the least-angle strategy. *Spatial Cognition and Computation 2,* 4, 283–313.

HUANG, W., EADES, P., AND HONG, S.-H. 2009. A graph reading behavior: Geodesic-path tendency. In *2009 IEEE Pacific Visualization Symposium*, P. Eades, T. Ertl, and H.-W. Shen, Eds. Institute of Electrical and Electronics Engineers, 137–144.

HUBLEY, A. M. AND ZUMBO, B. D. 2017. Response processes in the context of validity: Setting the stage. In *Understanding and investigating response processes in validation research*, B. D. Zumbo and A. M. Hubley, Eds. Vol. 69. Springer, 1–12.

KANE, M. AND MISLEVY, R. 2017. Validating score interpretations based on response processes. In *Validation of score meaning for the next generation of assessments*. Routledge, 11–24.

KANGASRÄÄSIÖ, A., JOKINEN, J. P., OULASVIRTA, A., HOWES, A., AND KASKI, S. 2019. Parameter inference for computational cognitive models with approximate bayesian computation. *Cognitive science 43,* 6, Article e12738.

LAMAR, M. M. 2014. *Models for understanding student thinking using data from complex computerized science tasks*. University of California, Berkeley.

LEIGHTON, J. P. AND GIERL, M. J. 2007. Defining and evaluating models of cognition used in educational measurement to make inferences about examinees' thinking processes. *Educational Measurement: Issues and Practice 26,* 2, 3–16.

LEVY, R. 2020. Implications of considering response process data for greater and lesser psychometrics. *Educational Assessment 25,* 3, 218–235.

LEVY, R. AND MISLEVY, R. J. 2017. *Bayesian Psychometric Modeling*. CRC Press.

LIU, H., LIU, Y., AND LI, M. 2018. Analysis of process data of pisa 2012 computer-based problem solving: Application of the modified multilevel mixture irt model. *Frontiers in Psychology 9*, Article 1372.

LUNDGREN, E. AND EKLÖF, H. 2020. Within-item response processes as indicators of test-taking effort and motivation. *Educational Research and Evaluation 26,* 5-6, 275–301.

MACGREGOR, J. N. AND CHU, Y. 2011. Human performance on the traveling salesman and related problems: A review. *The Journal of Problem Solving 3,* 2, Article 2.

MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. 2008. *Introduction to information retrieval*. Vol. 1. Cambridge University Press.

MOON, J. A., FINN, B., LAMAR, M., AND IRVIN R., K. 2018. Simulations of thought: The role of computational cognitive models in assessment. Periodical RDC-26, Educational Testing Service. September.

MUELLER, S. T., PERELMAN, B. S., AND SIMPKINS, B. G. 2013. Pathfinding in the cognitive map: Network models of mechanisms for search and planning. *Biologically Inspired Cognitive Architectures 5*, 94–111.

NETZEL, R., BURCH, M., AND WEISKOPF, D. 2014. Comparative eye tracking study on node-link visualizations of trajectories. *IEEE transactions on visualization and computer graphics 20,* 12, 2221–2230.

NEWELL, A. 1990. *Unified Theories of Cognition*. Harvard University Press, USA.

NEWELL, A. AND SIMON, H. A. 1972. *Human Problem Solving*. Prentice Hall.

OECD. 2013. *PISA 2012 Assessment and Analytical Framework*. https://doi.org/10.1787/9789264190511-en.

OECD. 2014. *PISA 2012 Results: Creative Problem Solving (Volume V)*. https://doi.org/10.1787/9789264208070-en.

PAASSEN, B., MCBROOM, J., JEFFRIES, B., KOPRINSKA, I., YACEF, K., ET AL. 2021. Mapping python programs to vectors using recursive neural encodings. *Journal of Educational Data Mining 13,* 3, 1–35.

PENG, F., SCHUURMANS, D., AND WANG, S. 2004. Augmenting naive bayes classifiers with statistical language models. *Information Retrieval 7,* 3, 317–345.

PYLYSHYN, Z. W. 1984. *Computation and cognition: Towards a foundation for cognitive science*. MIT Press.

QIAO, X. AND JIAO, H. 2018. Data mining techniques in analyzing process data: A didactic. *Frontiers in psychology 9*, Article 2231.

RAFFERTY, A. N. 2014. *Applying Probabilistic Models for Knowledge Diagnosis and Educational Game Design*. University of California, Berkeley.

RAFFERTY, A. N., ZAHARIA, M., AND GRIFFITHS, T. L. 2014. Optimally designing games for behavioural research. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 470*, Article 20130828.

RAHMANI, V. AND PELECHANO, N. 2022. Towards a human-like approach to path finding. *Computers & Graphics 102*, 164–174.

REITTER, D. AND LEBIERE, C. 2010. A cognitive model of spatial path-planning. *Computational and Mathematical Organization Theory 16,* 3, 220–245.

ROMERO, C. AND VENTURA, S. 2020. Educational data mining and learning analytics: An updated survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10,* 3, Article e1355.

ROSÉ, C. P., MCLAUGHLIN, E. A., LIU, R., AND KOEDINGER, K. R. 2019. Explanatory learner models: Why machine learning (alone) is not the answer. *British Journal of Educational Technology 50,* 6, 2943–2958.

RUSSELL, S. AND NORVIG, P. 2021. *Artificial Intelligence, Global Edition A Modern Approach*. Pearson.

SALLES, F., DOS SANTOS, R., AND KESKPAIK, S. 2020. When didactics meet data science: Process data analysis in large-scale mathematics assessment in france. *Large-scale Assessments in Education 8,* 1, 1–20.

STEWART, T. AND WEST, R. 2010. Testing for equivalence: a methodology for computational cognitive modelling. *Journal of Artificial General Intelligence 2,* 2, 69–87.

TANG, X., WANG, Z., LIU, J., AND YING, Z. 2021. An exploratory analysis of the latent structure of process data via action sequence autoencoders. *British Journal of Mathematical and Statistical Psychology 74,* 1, 1–33.

TURNER, A. 2009. The role of angularity in route choice. In *International Conference on Spatial Information Theory*, K. Stewart Hornsby, C. Claramunt, M. Denis, and G. Ligozat, Eds. Springer Berlin, Heidelberg, 489–504.

ULITZSCH, E., HE, Q., ULITZSCH, V., MOLTER, H., NICHTERLEIN, A., NIEDERMEIER, R., AND POHL, S. 2021. Combining clickstream analyses and graph-modeled data clustering for identifying common response processes. *psychometrika 86,* 1, 190–214.

WALSH, M. M., ARSLAN, B., AND FINN, B. 2021. Computational cognitive modeling of human calibration and validity response scoring for the graduate record examinations (gre). *Journal of Applied Research in Memory and Cognition 10,* 1, 143–154.

WARE, C., PURCHASE, H., COLPOYS, L., AND MCGILL, M. 2002. Cognitive measurements of graph aesthetics. *Information visualization 1,* 2, 103–110.

WIENER, J. M. AND MALLOT, H. A. 2003. 'fine-to-coarse'route planning and navigation in regionalized environments. *Spatial cognition and computation 3,* 4, 331–358.

XU, H., FANG, G., CHEN, Y., LIU, J., AND YING, Z. 2018. Latent class analysis of recurrent events in problem-solving items. *Applied Psychological Measurement 42,* 6, 478–498.

XU, H., FANG, G., AND YING, Z. 2020. A latent topic model with markov transition for process data. *British Journal of Mathematical and Statistical Psychology 73,* 3, 474–505.