Title: Automatic Scoring of Items in PISA 2012 Problem Solving

Author: Ray Philpot, Australian Council *for* Educational Research

Email: Ray.Philpot@acer.edu.au

Abstract:

The OECD PISA 2012 Problem Solving assessment was delivered on-line and exploited the capabilities of the computer by including many problems where students were required to interact with a virtual scenario. The actions taken by students – clicking buttons, dragging objects etc – were recorded in log files during testing. For many items these logs were used to automatically score the item. In some cases the strategy exhibited by the students as they tackled a problem contributed to the scoring, in addition to whether they managed to solve the problem.

In this paper I give an overview of the PISA 2012 Problem Solving assessment; describe the types of interactive problems used in the assessment; present sample items to illustrate how students interact with the problem scenarios; and explain how actions relating to strategies were captured, interpreted and scored automatically.

Key Words: Automatic scoring, Problem solving, Computer-based assessment

1. Problem Solving and the PISA 2012 assessment

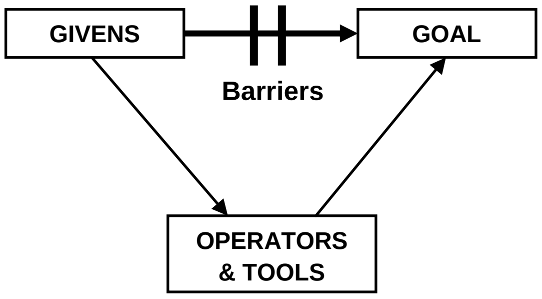
Mayer (1990) defines *problem solving* as cognitive processing directed at transforming a given situation into a goal situation when no obvious method of solution is available. Thus a problem must be novel to the "solver" (person attempting to solve the problem) in some respects.

Another way of looking at a problem is given by Frensch and Funke (1995), who represent a problem situation as in

Figure 1. Problem Situation

.

Figure 1. Problem Situation



Givens is the knowledge the person has about the problem at the outset. *Barriers* exist that must be overcome (e.g. lack of knowledge or obvious strategies) stand in the way of achieving the desired *goal*. *Operators* are the allowable actions that can be performed to achieve the goal with the assistance of the available *tools* (including strategies).

The PISA 2012 assessment of problem solving (OECD, 2013) used these ideas, but makes a distinction between "static" and "interactive problems". If all the information necessary to solve the problem is disclosed to the solver at the outset, the problem is called *static*; otherwise it is called *interactive*. An example of a static problem is the well-known Tower of Hanoi problem. Interactive problems are discussed in the next section.

The PISA 2012 Problem Solving assessment contained a mixture of static and interactive problems, all delivered by computer. As will be discussed at length in this paper, many problem solving items were computer-scored, both static and interactive.

2. Interactive problems

Interactive problems arise when the solver must explore the problem space (to use the terminology of Newell & Simon, 1972) to uncover information that is essential in achieving the goal. A common example of such a problem is working out how to use an unfamiliar electronic device such as a household appliance, mobile phone or ticket machine. In the absence of instructions or a manual, the user of the device must *experiment* with it to build up a model of how it works: it might not be possible to deduce the effect of pushing a button on a remote control, for example, without actually pushing it to see what happens.

It is often possible to implement interactive problems on a computer. Problems where the solver explores and controls a simulated environment are indeed a distinctive feature of the PISA 2012 Problem Solving assessment. Two popular ways of modelling interactive problem situations are by using Finite State Machines and Difference Equations or "MycroDYN" implementations (see Buchner and Funke, 1993 and Greiff and Funke, 2008 for background details).

Static problems can, of course, also be implemented on a computer. Whilst they can be presented on paper, using a computer offers many advantages including: the capability of presenting material in a more engaging way to test takers by using multimedia features such as animation; making online tools available; and using a wide range of response formats including clicking on objects and "drag and drop". Naturally these features are available for use when implementing interactive problems, too.

3. Sample items

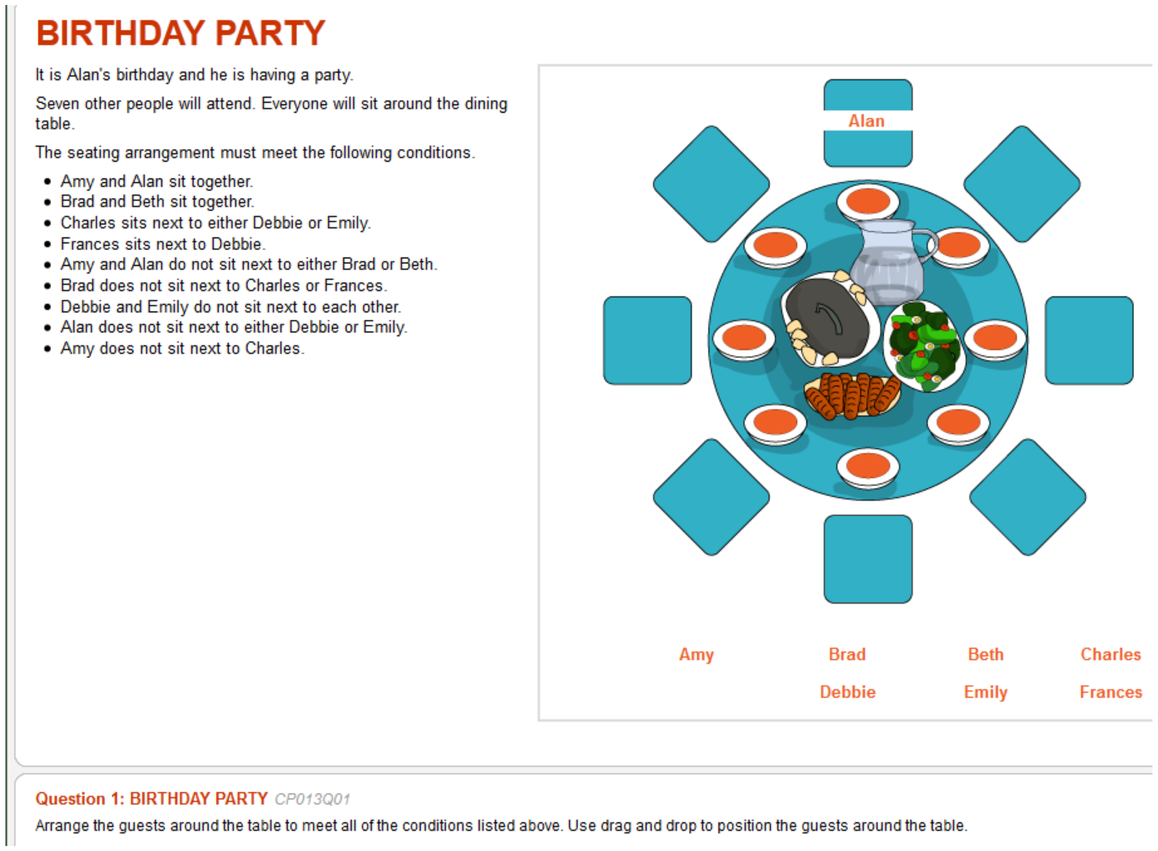
Two units released after the PISA field trial are based on arranging seating for guests around a table (Birthday Party) and exploring and controlling an unfamiliar electronic device (MP3 Player). These can be used to illustrate how static and interactive problem situations were implemented in PISA.

<u>Birthday Party</u>

A screenshot from an item from the unit Birthday Party is shown in *Figure 2*. The problem here is to seat people around a table, subject to a number of constraints. This is a static problem, in the sense that all the information necessary to solve the problem – in particular, the names of the people, the available seating and the constraints to be satisfied – is given. Students can drag names onto seats, and indeed this is how they provide a solution.

Automatic scoring for this item is done based on the final configuration of the seating, using information from the log files; see Section 4 below for how log files are used. Full credit is awarded if all constraints are satisfied; partial credit is awarded if all but one of the constraints is satisfied. Note that in this item the *strategy* used by the solver does not play a part in the scoring.

Figure 2. Birthday Party screenshot


BIRTHDAY PARTY

It is Alan's birthday and he is having a party.

Seven other people will attend. Everyone will sit around the dining table.

The seating arrangement must meet the following conditions.

- Amy and Alan sit together.
- Brad and Beth sit together.
- Charles sits next to either Debbie or Emily.
- Frances sits next to Debbie.
- Amy and Alan do not sit next to either Brad or Beth.
- Brad does not sit next to Charles or Frances.
- Debbie and Emily do not sit next to each other.
- Alan does not sit next to either Debbie or Emily.
- Amy does not sit next to Charles.

Question 1: BIRTHDAY PARTY CP013Q01

Arrange the guests around the table to meet all of the conditions listed above. Use drag and drop to position the guests around the table.

<u>MP3 Player</u>

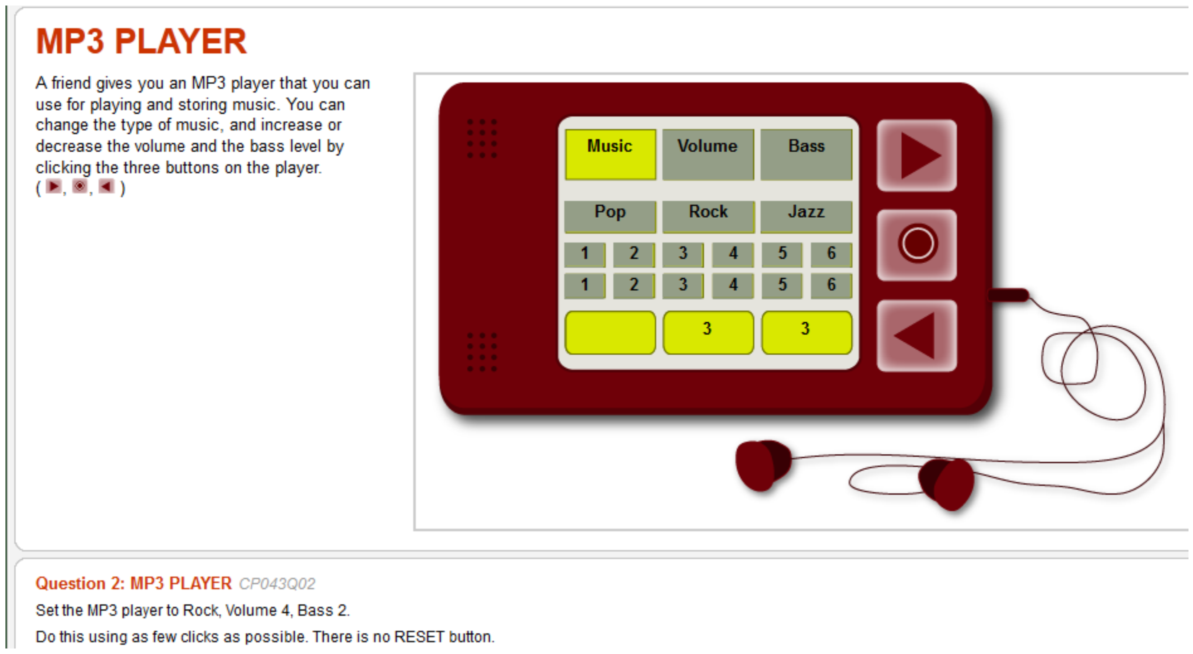
A screenshot from an item from the unit MP3 player is shown in *Figure 3*. The underlying structure of the MP3 player is based on a Finite State Machine. The particular interface presented to the solver is unfamiliar. The problem here is to place the MP3 player in a particular state, using as few clicks as possible. In a previous item, the solver has had a chance to explore how the MP3 player works, with no restriction on how many clicks are used.

Automatic scoring for this item is done based on the final settings on the player and the number of clicks recorded, using information from the log files. Full credit is awarded if the MP3 player is placed in the correct configuration in at most 13 clicks. In fact the minimum possible number of clicks required is 11, so a mistake or slightly non-optimal path to the solution can still attract full credit. Nevertheless, only a highly efficient strategy will enable the solver to obtain full credit.

Partial credit is awarded for the correct configuration obtained in *more than* 13 clicks, with no upper bound. Here persistence is rewarded, but with a penalty for inefficiency.

As can be seen, in this item there is a crude identification of strategy, which is captured in the different scoring categories.

Figure 3. MP3 Player screenshot


MP3 PLAYER

A friend gives you an MP3 player that you can use for playing and storing music. You can change the type of music, and increase or decrease the volume and the bass level by clicking the three buttons on the player. (▶, ●, ◀)

Question 2: MP3 PLAYER CP043Q02

Set the MP3 player to Rock, Volume 4, Bass 2.

Do this using as few clicks as possible. There is no RESET button.

Note: The two units "MP3 Player" and "Birthday Party" are available for viewing on the web at cbasq.acer.edu.au, using the credentials "public" and "access".

4. Capturing interactions

As noted above, automatic scoring of items is made possible through the use of log files. Before discussing scoring further, some general remarks on the *benefits* of capturing process data and the *types* of process data that can be captured will be made. This will help in understanding how automatic scoring can be done, in particular how strategic behaviour can be attributed to solvers in view of their interactions with the computer.

Process data can be used for scoring of items, leading to reporting of student proficiency based on the analysis of scores. It can additionally be used for profiling student behaviour, provided significant performance features can be identified in the voluminous data obtained.

Of particular interest for this sort of analysis are things such as: the *sequence* of certain events; *timing* per action/item/unit/test form; and *state* of system variables at various points during the course of an item. Examples of pertinent junctures include:

- at the point the student exits the item;
- after a specified time interval; and
- upon entering or leaving a given system state.

In more detail, the types of data captured can include:

- Type of interaction
  - ➢ Mouse click
  - ➢ Mouse down/up/rollover
  - ➢ Drag and drop
  - ➢ Focus/Blur on control
  - ➢ Keystrokes
  - ➢ Dropdown item/option clicked on

- Frequency of interaction
- Time between events, including time to first event
- Eye movement, heart rate, skin conductivity *etc*
- Custom events (defined beforehand by the test developer)
- Values of system variables such as:
  - ➢ Position of the mouse cursor within a coordinate system when an interaction was made (e.g. a click)
  - ➢ Positions of on-screen objects "dragged" by the user
  - ➢ Values of on-screen numeric variables manipulated by the user
  - ➢ Values of hidden or read-only numeric variables

All of the above – with the sole exception of eye movements *etc* – were captured in the XML-based log files collected during the PISA 2012 Problem Solving assessment.

An example of a custom event that could be defined is when a user moves from cell (4,6) to cell (3,6) within a predefined two-dimensional grid structure (e.g. moving through an on-screen rectangular maze). Rather than having to interpret a series of mouse clicks and internal variable changes, a custom event can be defined that produces an XML element such as

<CP999_path><![CDATA[(x:4,y:6)->(x:3,y:6)]]></CP999_path>.

This can be used directly in scoring, as will be seen in the next section.

5. Automatic scoring

With the exception of a few expert-coded, open constructed response items, the PISA 2012 Problem Solving items were automatically scored. In theory this could take place at the time the item was responded to, but in fact the scoring was done centrally at a later time to allow data cleansing to have taken place.

Naturally multiple choice items and items requiring a single numeric response can be automatically scored. Of more interest are items where the solver interacts with the computer-simulated problem situation, performing multiple steps in order to reach a solution or goal state. For these items, scoring algorithms were written based on events to be captured in the XML log files.

Logically, events can be mapped to system features or states such as:

- How often a button was clicked (if at all)
- Whether the system reached a particular state (for example, in the MP3 Player – a Finite State Machine – whether the volume was set to 3)
- Whether endogenous (internal) variables lie within a particular range (for example whether the temperature was set to be within 21 and 23 degrees)
- What values of exogenous (user-input) variables were chosen
- Whether on-screen objects lie within pre-defined regions (for example, whether users have dragged a block onto a square in a board game)
- Whether particular nodes, links, table cells *etc* are highlighted (for example, whether a route on a map has been selected)

As an example of what goes into a scoring algorithm, consider a problem involving traversing a two-dimensional integer grid (subject to constraints), as mentioned in the previous section. If one of the solution requirements is to pass from the node (4,6) to the node

(3,6), then the scoring algorithm can use an XML query to check for the presence of the value (x:4,y:6)->(x:3,y:6) in the <CP999_path> elements in the log file.

To take another example, each button click made in the MP3 player item discussed earlier was recorded in the log file as a special event. An XML query is used to tally the number of such events present in the log file. The final state of each of the variables is also queried. Full credit can then be awarded if the MP3 player has been placed in the correct configuration in at most 13 clicks.

The discussion so far has not been directly concerned with strategies. How are user behaviours to be captured and interpreted for use in scoring?

6. Interpreting and using strategies for scoring

System states and features can in some cases be mapped to general behaviours that can be used in scoring items. Examples of behaviours that can be captured in this way are:

- **Persistence**: how many failures before success was attained
- **Thoroughness**: how much exploration was performed and how much information was uncovered in the process
- **Learning**: how much knowledge (factual, conceptual, procedural) was acquired
- **Efficiency**: how systematic and efficient the solver was (how close to an optimal method of solution)
- **Quality of system control**: how well was the solver able to control a dynamic system
- **Strategy**: whether a strategy used, for example, VOTAT (= Vary One Thing At a Time) or trial and error

Most of these behaviours were used for scoring in PISA 20120 Problem Solving. Some examples are given below.

Learning was captured in items based around so-called MicroDYN systems (modelled using difference equations: see Blech and Funke, 2005, and See Greiff and Funke, 2008). A typical example is a computer-simulated scenario where there are three different types of flowers in a garden and three different fertilisers are available to use. One task is to work out which fertiliser(s) affects the growth of which flower(s). Solvers can try different amounts of any or all of the fertilisers in a “round” and once the fertilisers are applied, the effect on the flowers is shown (numerically or graphically). *Learning* is captured by asking solvers to draw the relationship between the fertilisers and flowers.

Figure 4. Typical MicroDYN relationship map

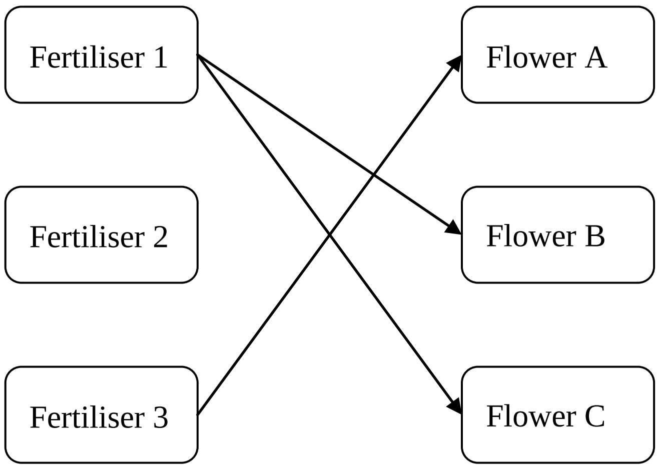


Figure 4 shows the sort of relationship map that might be drawn by solvers: they drag arrows between the fertiliser and flowers to create the map. It can be seen from *Figure 4* that Fertiliser 1 affects Flowers B and C; Fertiliser 3 affects Flower A; and Fertiliser 2 doesn't affect any type of flower. Scoring of learning is done automatically, based on the accuracy of the arrows.

To check whether the students were using an effective *strategy* in their experimentation, the log files were analysed to check whether there were rounds with no activity (to check if the flowers grow without fertiliser) and whether there were rounds where only one fertiliser was used (whether the VOTAT = Vary One Thing At a Time strategy was used). Scoring was computed automatically based on this information.

In subsequent items in these types of scenarios, *quality of system control* was tested by giving solvers a target growth (generally a range of values) for each flower and a limited number of rounds to achieve these targets. The numerical distance from the goal states was used to automatically compute a score for quality of system control.

Thoroughness was tested in the context of buying tickets from an automated ticketing machine (modelled as a Finite State Machine). Automatic scoring for full and partial credit was based on the final choice of ticket made *and* the extent of exploration performed: did the solver explore enough to be able to <u>know</u> that the best choice of ticket was made?

In a number of items there were many solution paths of different lengths/steps. *Efficiency* was measured by the number of steps used to reach a solution, the fewer the better. The example cited earlier of the MP3 Player illustrates this.

7. Conclusion

Zoanetti (2010) has had success demonstrating the relationship between process data and expected problem-solving behaviours in simple "static" puzzle-type problems (solved by search-based heuristics). He used the results of think-aloud protocols to match events to student problem solving behaviours such as whether or not solvers repeat ineffective actions.

As illustrated in this paper, it is possible *using process data alone* to capture and interpret actions relating to complex problem solving strategies such as VOTAT, arising during the course of solving dynamic, interactive problems. This information can then be used to automatically score responses to items, ultimately leading to precise measurement of individuals' problem solving abilities.

It might be fruitful to combine the approach described in this paper with Zoanetti's work, with the aim of refining the criteria by which good problem solving can be identified from process data.

8. References

Blech, C. and J. Funke (2005), Dynamis review: An overview about applications of the Dynamis approach in cognitive psychology, Deutsches Institut fűr Erwachsenenbildung, Bonn, http://www.die-bonn.de/esprid/dokumente/doc-2005/blech05_01.pdf.

Buchner, A. and J. Funke (1993), Finite-state automata: Dynamic task environments in problem-solving research, *The Quarterly Journal of Experimental Psychology*, Vol. 46A, No. 1, pp. 83–118.

Greiff, S. and J. Funke (2008), *Indikatoren der Problemlöseleistung: Sinn und Unsinn verschiedener Berechnungsvorschriften. Bericht aus dem MicroDYN Projekt [Measuring Complex Problem Solving: The MicroDYN approach]*, Psychologisches Institut, Heidelberg.

Mayer, R.E. (1990), "Problem solving", in M. W. Eysenck (ed.), *The Blackwell Dictionary of Cognitive Psychology* , Basil Blackwell, Oxford, pp. 284–288.

Newell, A., & Simon, H.A. (1972), *Human Problem solving*. Englewood Cliffs, NJ:Prentice-Hall, Inc.

OECD (2013), *PISA 2012 Assessment and Analytical Framework: Mathematics, Reading, Science, Problem Solving and Financial Literacy*, OECD Publishing, Paris. http://dx.doi.org/10.1787/9789264190511-en

Zoanetti, N. (2010), Interactive computer based assessment tasks: How problem-solving process data can inform instruction. *Australasian Journal of Educational Technology* 2010, 26(5), 585-606.