Mei:CogSci
University of Vienna

Summer semester 2017

Masters project proposition

Problem Solving Perspectives

Nejc Grenc

March 16, 2017

"You may shoe the horse, feed it only the best wheat and even motivate it with a carrot, but if you do not replace the skies on your carriage with wheels, it will hardly move."

Increasing amount of research in the field of Problem Solving is dedicated to *optimization*. This is inherently good, as optimization ensures the problem-solving process is concluded faster, with less resources and/or the produced solution is better. However, optimization can achieve only so much. Often a simple *change of perspective* can produce massive improvement that overshadows any optimization.*Perspective change* does, of course, not work for all problems and is inherently very hard to accomplish, but as its improvements are enormous, it is worth researching.

Contents

In case of further interest, author can be contacted via e-mail: nejc.grenc@gmail.com

1 Introduction

The general idea of Problem Solving is to provide solutions to specified problems. This is a very broad definition and as this field has been widely researched through many disciplines (most notably psychology, medicine, engineering, computer science, artificial intelligence and mathematics) finer details of Problem Solving have been discovered. There is a wide area of problems, some of which are well- or ill- defined, they differ in complexity, transparency, dynamics and multiplicity of goals [1]. The process in which one tackles a given problem (problem-solving process) can differ a lot as well, depending on the strategy one uses [2]. These are one of the most researched aspects of Problem Solving. Unfortunately, one of the most important aspects is often considered intuitive and hereby very neglected: Shape of the problem and problem-solving process. Given a problem definition, the problem is far from uniquely defined. The system's internal representation of the problem (partially depending on prior knowledge and problem's description) then fully shapes the problem (and solution) spaces and consequently the problem-solving process as well. This problem and problem-solving-process' general shape determines how problem-solving will occur, which optimizations are feasible and which strategies can be applied. A single problem (and problem-solving-process') shape is hereby marked as a *perspective*.

It is not clear if all problems are able to partake in multiple perspectives, as at the moment we do not know what properties enable a specific problem to do so. But in case at least two perspectives of a problem have been identified, it is easy to imagine the differences they cause among the problem-solving processes; the time and resource consumption may wary as well as solution's quality or optimality (presuming the solution is achieved in both cases). Unfortunately, even if a problem has multiple perspectives, it can be very difficult to identify them.

1.1 Multiple equation example

Mathematics is full of simple examples that show us how perspective can influence the way of solving a problem. One of notable examples is *the multiple equation example*:
The task is to solve the fourth equation:

$$\begin{array}{c} a+b=7 \\ a+c=11 \\ b+c=10 \\ \hline a+b+c=\,? \end{array}$$

The common (roughly 97% of times) attempt to solve this problem is to calculate the values of each variable by combining all of the equations, like so: $a = 7 - (10 - (11 - a))$; $a = 4$. After calculating each of the variable values, the fourth equation is simple to solve. This perspective focuses around idea of *variables as unknown values that need to be solved* and can be commonly found being thought in primary and secondary schools. It requires a lot of computations, but it gives us all of the primary values of this problem (though this data can be useful, it is not necessary for this problem).

When attempting to relieve themselves of the burden to compute all variable values, some solvers combine equations together in order to reach the fourth equation form, like so: $a+b+c = ((a+b)+(a+c)+(b+c))/2 = (7+11+10)/2$. This approach is strongly preferred according to computational complexity, as it requires the minimum number of mathematical functions to be solved; two additions and a division. To view the problem through this perspective can be extremely beneficial, but it is very rarely observed.

2 Preceding research

In the winter semester 2016/2017 on my Erasmus+ exchange in Vienna, I have conducted a project under the mentorship of Dr. Paolo Petta (from the Austrian Research Institute for Artificial Intelligence). The project was titled: *"The Effect of Programming Paradigms on Problem Solving"* and it tried to answer the research question: "How do programming paradigms affect how humans tackle problems". To put it shortly, the main focus of the project was to determine the aspects of programming paradigms that influence human problem-solving process and to determine the effects of these aspects.

During the first phase of the project literature was collected and comprehended. Of significance was the literature from the psychological point of human problem solving, the computer-science's view of logic programming and programming/engineering standpoint on code analysis. After this phase, a detailed review of two selected programming paradigms was formed in order to better understand them and determine the aspects in which they differ. The selected paradigms were: *(1) Object-oriented programming paradigm* for its prevalence in programming community and *(2) Logic programming paradigm* for its unique approach to programming (and still being reasonably known). These two were thoroughly examined during the course of the project.

During the course of the project a short pilot experiment was conducted, in which subjects were asked to write a code that will solve the Tower of Hanoi problem. They were asked to write it for each selected programming paradigm in their respectable programming languages (for (1) language Java was selected and for (2) Prolog). The experiment procedure used was Think Aloud [3]. This experiment provided some critical insight on our research, as the thinking process and code produced for each perspective differed substantially.

2.1 Tower of Hanoi experiment

This is a simple puzzle, containing a few (usually 3) disks of different sizes and 3 poles. The objective of the puzzle is to move the entire stack from the starting pole to the goal pole, obeying the following simple rules: (1) Only one disk can be moved at a time. (2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack. (3) No disk may be placed on top of a smaller disk.

When undergoing the Think Aloud [3] experiment in the perspective of Object-oriented programming paradigm (using the language Java) they firstly set up the internal representation of the problem space using programmable objects. Afterwards, some form of graph theory was implemented - stable states were identified (positions of disks on poles) and moves between them. As the problem space was set up, some search algorithm (usually depth search) was executed over the problem graph, that produced the end solution. Solutions obtained this way were sub-optimal, but the physical representation was very clear and though the code was long and it took long time to produce, it was easily readable.

When undergoing the same experiment from the perspective of Logic programming paradigm (using the language Prolog), the initial process of designing the code took longer, but the whole programming process was concluded in around a quarter of time used in the other perspective. The same goes for the code length. The idea subjects used here was double recursion (to move unwanted disks away) and a move of the desired disk. This procedure produces an optimal solution in an shorter period of time and with less written code compared to the Object-oriented perspective, however the code is much harder to read.

This pilot study has provided us with evidence that under different perspectives, the problem-solving processes and solutions can differ substantially.

With the help of our experiment, some crucial 'ideas' were identified in the problem-solving processes. These 'ideas' were marked as ***archetypes*** and they play a crucial role in the shaping of a problem-solving process in a specific perspective. In the Object-oriented programming paradigm, an *Object-representation* (using mental images of physical objects), *Graph-space* (representing the problem in form of "stable problem states" and actions to move between them, preferably in a graph visualization) and *Depth-search archetypes* (searching through move sequences in a deepening manner;in contrast to breadth-search) were mostly observed. On the other hand, in the Logic programming paradigm the archetypes used were mostly: *Sub-problems* (splitting problem into smaller problems) and *Recursion* (using the same function inside the function). Each of these archetypes provides unique design (of problem-solving process) and solution features, that may or may not be desirable, and a set of these archetypes fully defines a perspective.

Support for our theory can be found in works of Kotovsky K. and Fallside D. (former associates of Herbert A. Simon) [4] where they observed difference in internal representation of a problem. Subjects were presented with a changing ball on a computer screen - one group was told the ball is increasing / decreasing in size ("size" problem) while the other was told that the ball is coming closer / retreating further ("distance" problem); while both groups were observing the same stimulus. The internal representation subjects made had strong impact on their problem-solving process and transfer (of knowledge) [1], as subjects who were solving the "size" problem preformed better and their knowledge was better transferred, compared to subjects who were solving the "distance" problem. Their research can be easily applied to our archetype model; archetypes are transferable entities that contain some individual features, which are ultimately reflected in the problem-solving process.

A lot of research has already been done on identifying the perspectives, archetypes and the features they produce. However, a lot more is required to identify all of them and to fully understand their properties and their effects on final solutions.

3 Master thesis

In my master thesis, I would like to continue working on this project. Firstly, I plan to expand this model by examining even more problems with at least two perspectives identified, and in this way catalog common archetypes. According to my background knowledge and experience it makes sense to first tackle the problems in fields of mathematics and computing in my search for archetypes and afterwards envelop other scientific fields. After the list of archetypes is complete, I plan to dive into each of them and research their properties, features and their function in problem perspectives. In this way, we will be able to get an insight on how archetypes and perspectives work, which will be of crucial help when we attempt to construct new perspectives in problems, which have yet to have more fully identified perspectives. Finally, the finished model can be taught to subjects and in this way increase their ability of perspective identification and improve their problem-solving skills on a general level.

[1] "Transfer" marks the application of knowledge, gained from solving a specific problem, on a novel problem and thus improve its problem-solving process. [4]

3.1 Project plan draft

(1) Defining the project
Set up the project plan and get it approved by mentor.
Set up project workspace, preferably not at home.
Set up project wiki for logging the work and all of the collected data.
Estimated work time: 20 hours

(2) Acquisition of problems
Search for problems which have at least 2 identified perspectives and catalog them.
Estimated work time: 60 hours

(3) Setting up the experiment
Select a few cataloged problems.
Create an experiment template.
Invite participants to solve problems while being monitored.
Estimated work time: 40 hours

(4) Running the experiment
Participant solves the problem while explaining his/hers every thought.
The transcript and solution are afterwards analyzed.
Estimated work time per participant: 5 hours (experiment) + 10 hours (analysis)

(5) Final analysis
Analysis of the gathered data and integrating it into the existing knowledge.
Expand the current archetype model or create a new model. Estimated work time: 80 hours

(6) Report and thesis
Writing the final report and thesis.
Estimated work time: 100 hours

This project plan is just a draft. It will be fixed and expanded after meeting with mentor.

Requirements

I require a mentor and a place to work and conduct the experiments.
Everything else will be provided by me.

3.2 Mentor

This project requires a mentor!

As I can provide knowledge on mathematics, computing and problem-solving, it would be great if the mentor has in-depth knowledge about **human psychology** and some knowledge about general problem-solving and project (and experiment) design. Otherwise, anyone really interested in this project is gladly welcome to participate in any way he/she can.

This project has great potential to provide insight on the world of problem perspectives. Each perspective has its own quirks, side effects, solution features and other features, that are yet to be discovered, and to understand them can provide a crucial help in selecting the optimal one for each problem. Preferably, we will also be able to understand how to initiate a perspective change and in this way create/discover new perspectives. This way, we will create a framework, that enables us to always select 'the best tool for the job' - in this regard, the best perspective. Establishing such a gateway to the world of perspectives is an amazingly huge step towards general understanding and conquering problems.

"Change the way you look at things and the things you look at change." (W. Dyer)

References

[1] Dietrich Dörner and A Wearing. Complex problem solving: Toward a (computer-simulated) theory. *Complex problem solving: The European perspective*, pages 65–99, 1995.

[2] Yingxu Wang and Vincent Chiew. On the cognitive process of human problem solving. *Cognitive Systems Research*, 11(1):81–92, 2010.

[3] Clayton Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center, 1982.

[4] Kenneth Kotovsky and David Fallside. Representation and transfer in problem solving. *Complex information processing: The impact of Herbert A. Simon*, pages 69–108, 1989.