Adapting Between Parsons Problems and Coding Tasks

Nadia L. Goralski • **Jeremy S. Bradbury**
Faculty of Science • **Ontario Tech University** • **Oshawa, ON, Canada**
nadia.goralski@ontariotechu.net, jeremy.bradbury@ontariotechu.ca

1. Motivation

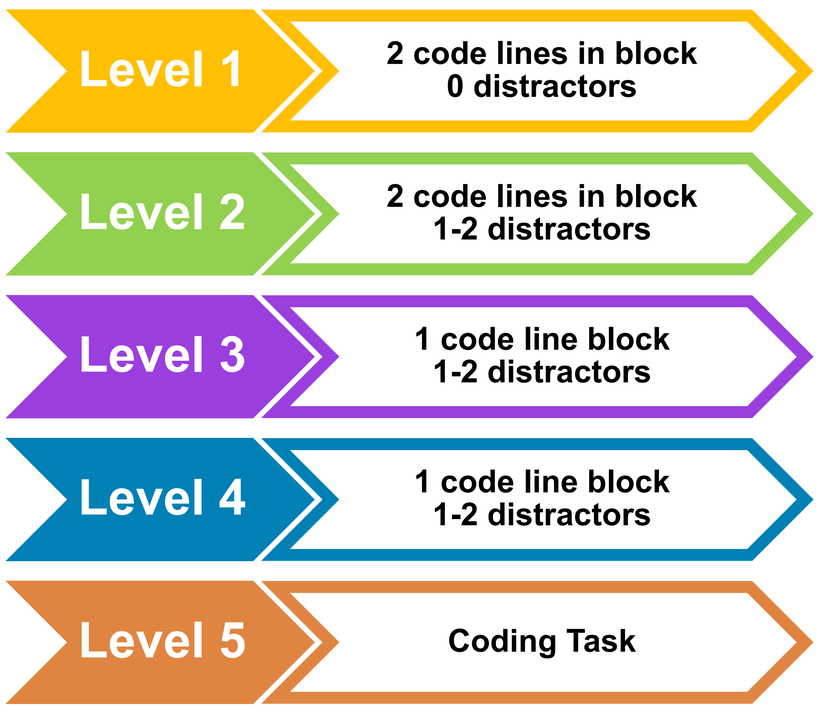
- Parsons problems are an effective scaffolding technique for learning coding [3].
- However, Parsons problems are often viewed as a separate learning activity from coding tasks.

RESEARCH GOAL:
Create an adaptive educational Parsons problem tool that incorporates writing code and allows the learner to adjust the difficulty based on their skill level.

2. Background

- **Parsons problems** involve students rearranging blocks of code to form a correct solution [4].
- **Distractors** are incorrect block of code that are incorporated into a Parsons problem to increase the difficulty [2].
- **Adaptive Parsons problems** allow the learner to adjust different parameters of the problem such as the size of code blocks or the number of distractors [3].

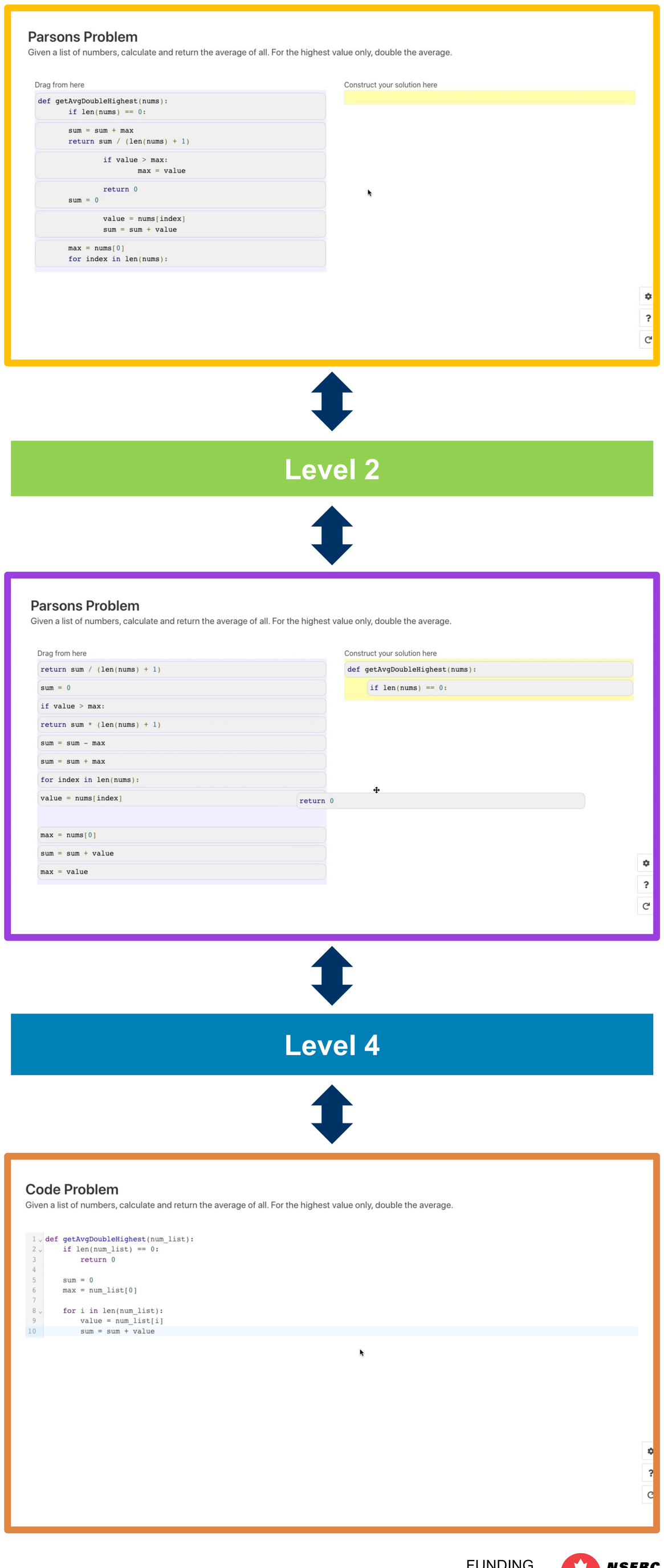
3. Level Design



4. Transitioning Code to Parsons Problem

- It is considerably more challenging to adapt between a Parsons problem and source code then it is to adapt the level of difficult within a Parsons problem.
  - This difficulty is primarily due to the free-form nature of coding.
  - In a coding task different leaners may use different variable names and make different algorithmic choices while in a Parsons problem the elements of the solution are fixed.
- Bridging the gap from a Parsons problem to a coding task is straightforward and involves presenting any partial Parsons problem solution as source code in a coding editor.
- Bridging the gap from a coding task to a corresponding Parsons problem is considerably more challenging and involves several steps:
  - Clone detection (via the NiCad clone detection tool [1]) is first used to map code fragments in a a student coding task solution to blocks in a Parsons problem solution.
  - Once student code fragments are mapped, the variable names in the student code are preserved and replace any variable names in the Parsons problem solution.
  - Any missing parts of the Parsons problem solution are listed as unused blocks.
  - Any learner generated mistakes can be included as distractors.

5. Tool Overview



6. Future Work

- Conduct a user study in a first-year university class to assess the learning benefits of our approach
- **Research Questions:**
  1. How do students transition from a Parsons problem to a coding task?
  2. If a student struggles with a coding task is it valuable to transition back to a Parsons problem?

7. References

[1] James R Cordy and Chanchal K Roy. 2011. The NiCad clone detector. In *Proc. of the 19th Int. Conf. on Program Comprehension (ICPC 2011)*. IEEE, 219–220.
[2] Yuemeng Du, Andrew Luxton-Reilly, and Paul Denny. 2020. A review of research on Parsons problems. In *Proc. of the 22nd Australasian Computing Education Conference (ACE'20)*. 195–202.
[3] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the affect and effect of adaptive Parsons problems. In *Proc. of the 19th Koli Calling Int. Conf. on Computing Education Research (Koli Calling '19)*. 1–10.
[4] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proc. of the 8th Australasian Conference on Computing Education-Volume 52 (ACE'06)*. 157–163.

SIGCSE 2023 Technical Symposium
SIGCSE • Toronto, ON, Canada • March 2023
FUNDING PROVIDED BY: NSERC CRSNG