The paper "Extracting Features for Computational Thinking from Block-Based Code" by Param Damle, Jo Watts, Glen Bull, and N. Rich Nguyen explores the development of an automated evaluator (autograder) for Snap!, a block-based programming language. This evaluator aims to support undertrained instructors in introductory computer science classes by assessing the computational thinking (CT) demonstrated in the structure of students' code submissions rather than focusing solely on the accuracy or run-time efficiency of the code. The authors propose novel feature extraction methods tailored to capture essential CT elements such as abstraction and iteration from the XML tree representation of Snap! programs.

The study highlights the limitations of existing autograding methods, which are typically designed for text-based programming languages and often fail to capture the specific CT skills relevant to block-based coding environments like Snap!. To address this, the authors designed features that measure repetition and encapsulation patterns within the code, utilizing techniques such as a 1-dimensional convolution for repetition detection. They then applied various clustering algorithms, including K-means, DBSCAN, Gaussian Mixture Models, and hierarchical clustering, to group similar code submissions and evaluate the effectiveness of their feature extraction in reflecting CT principles.

Experimental results indicate that Gaussian Mixture Models and hierarchical clustering with two or three clusters showed promise in effectively grouping student submissions based on CT elements, achieving near-optimal relative inertia scores. The study concludes with an acknowledgment of the potential for further tuning of feature extraction methods and clustering models to improve the autograder's performance. The authors also emphasize the importance of involving the academic community in the development and refinement of such tools to enhance scalability and applicability in diverse classroom settings.