Does Deliberately Failing Improve Learning in Introductory Computer Science?

Sverrir Thorgeirsson, Tanmay Sinha, Felix Friedrich, Zhendong Su

ETH Zürich, Switzerland

Abstract. We report our experience with technology-enhanced Productive Failure (PF) in an introductory computer science course. First, we sought to assess whether the use of algorithm visualization tools during the PF problem-solving phase enhanced learning. Second, we used an experimental study to measure learning effects of administering failure-driven scaffolding (FDS) during the PF sessions, that is, explicitly nudging generation with suboptimal representations deliberately designed to lead to failures. Results from surveys and log data indicated that our visualization tools helped students explore the problem space and performance data signaled that FDS improved students' constructive reasoning (Cohen's d 0.194, BF_{01} 2.55) and did not harm posttest scores (BF_{01} 3.17) relative to no explicit scaffolding during problem-solving prior to instruction. Further, similar levels of induced frustration (BF_{01} 3.29) and curiosity (BF_{01} 3.27) were observed across the conditions.

Keywords: active learning, failure, problem-solving, scaffolding

1 Introduction

Pedagogical activities inspired by active learning approaches are both expected and well-received by computer science undergraduates [1], and have a positive effect on their learning outcomes [3]. Within that family, a constructivist learning design called *productive failure* (PF) has received significant attention in the literature [5]. PF sessions consist of a (i) *problem-solving phase*, where students are given challenging problems that they are not expected to solve successfully, followed by an (ii) *instruction phase*, where an instructor illustrates the correct solution. Meta-analyses (e.g., [9]) show strong evidence in favor of PF and more generally, learning designs where problem-solving precedes instruction.

PF was initially described as a learning design with no explicit scaffolding. However, there has been recent interest in similar designs that incorporate *failure-driven scaffolding* (FDS), where students' self-generation activity in the problem-solving phase is complemented with prompts that nudge them to generate and reason with additional suboptimal numerical and graphical representations resulting in problem-solving failure by design. Comparative studies in tertiary data science education have shown that students receiving FDS demonstrate higher quality of constructive reasoning [8, 10], that is, provide meaningful elaborations going beyond what was presented. By challenging understanding,

FDS may help students activate relevant prior knowledge, reveal knowledge gaps, and aid recognition of deep domain features. However, despite holding promise, this learning design has not yet been explored in computer science.

Here, we report application of PF and FDS in a CS2 course, *Computer Science II*, with just over a hundred enrolled students. This course introduces algorithms and data structures to non-CS majors in the engineering departments of ETH Zürich. Throughout the course, we conducted technology-enhanced PF sessions which incorporated (i) online programming environment with an integrated testing and debugging suite, which was also used for homework assignments, and (ii) custom, interactive algorithm visualization (AV) environments designed to support an alternative and a more inclusive form of domain exploration. We focused on following three research questions (RQs) in our work.

RQ1 How does the provision of FDS during PF sessions impact learning outcomes of conceptual understanding and constructive reasoning for CS2 students?
- **H1** Students receiving FDS during PF sessions will demonstrate better learning outcomes of conceptual understanding and constructive reasoning (compared to students who do not receive FDS).

RQ2 What is the impact of providing interactive AV environment to CS2 students during the problem-solving phase of PF sessions (with or without FDS)?
- **H2** Students working with the interactive AV environment will have positive perceptions about its usefulness in facilitating problem-solving.

RQ3 How do affective factors differentially facilitate learning from FDS in the PF sessions for CS2 students?
- **H3** Students receiving FDS within PF will demonstrate higher frustration and discomfort as well as higher curiosity to know more about the topic.

2 Method

2.1 Study Design

After ethics approval and informed consent, we ran an experimental study on three PF sessions in an introductory computer science course in 2021 ($N = 64$, $n = 28$ female). All sessions were run remotely. In the problem-solving phase of each session, students were asked to devise their own algorithm to solve a problem that they had not encountered before in class. The problems involved (i) sorting numbers by size, (ii) finding shortest paths in a graph, and (iii) solving the cluster assignment problem. We chose these problems for two reasons. First, their discovery implies or requires insight into the key concepts covered in the course. Second, they follow design principles described by Kapur and Bielaczyc [5], namely that they have multiple solution paths and are sufficiently rich to allow "explanation and elaboration" and "compare and contrast" activities during the instruction phase between the canonical solution and student solutions.

In contrast to previous pilot PF sessions, we placed special emphasis on the *PF design fidelity* [9]. For example, we constructed interactive, visual environments (see Figure 1) for each problem that students could use to explore

the problem space and generate multiple solutions, despite lacking coding proficiency. For the session on sorting algorithms, a custom visualization based on the visual programming environment *Algot* [11] was used, which allowed students to demonstrate algorithms under the programming-by-demonstration paradigm. The remaining AVs can be viewed at http://sverrir.helonia.com/pfvis. By introducing AVs, we also hoped to make use of the dual coding framework emphasizing that AVs can be effective when presented together with code as they provide an additional, non-verbal model of the target knowledge [4], thereby offering learners a deeper domain understanding. Our aim was for these AVs to rank well on the AV engagement taxonomy introduced by Naps et al. [7]. To improve *PF design fidelity* further, we introduced failure-driven scaffolds during the problem-solving phase and measured how deliberately designed failure affected learning. Finally, we designed an appropriate social surround emphasizing that the PF sessions were an opportunity to learn and arriving at a correct solution was not the goal, and spent greater instruction time on explaining how the correct solution relates to student-generated solutions.

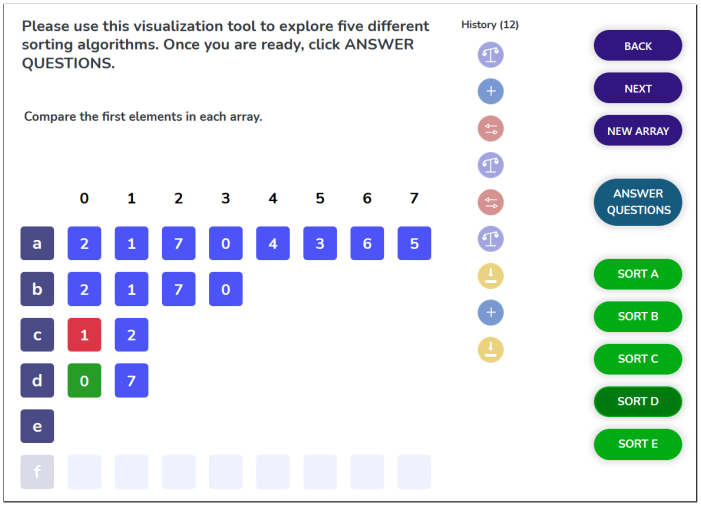

Fig. 1. The visualization environment introduced in the first PF session on sorting algorithms, which is based on the visual programming environment Algot [11].

To measure the effects of FDS, we set up an in-vivo experiment in which some students were randomly assigned FDS during the problem-solving phase, while others were given no explicit scaffolds at all and therefore engaged only in free generation prior to instruction. For example, during the second session on the shortest path problem, all students were given a Python implementation of a graph, preliminary code, and some test cases. However, students in the FDS group were suggested to model their solution based on an implementation of a relatively suboptimal solution (depth-first search), thus making it more challenging to reach the canonical solution when compared to, for example, breadth-first search. Similarly, during the third session on the clustering problem, all students

were also given some cues with Python functions that they could use to reach a solution. Students in the FDS group were however given additional functions that they were prompted to use (e.g., `nearest_neighbor_class`), which is not used in the k-means clustering algorithm, and therefore, by design, would likely lead to a suboptimal solution. Taken together, randomly assigning instructional treatments during the problem-solving phase (for high internal validity) as they occur in live classrooms (for high ecological validity), is a strength of our work.

The follow-up instruction phase was identical for both conditions. Posttest questions focused on conceptual understanding of the targeted topics and constructive reasoning. For example, one posttest question introduced the *widest path problem* and the *longest path problem* and asked whether and how Dijkstra's algorithm, which had meanwhile been taught during the lecture, could be modified to solve them. Similarly, another posttest question included a version of the anti-clustering problem, which has a similar conceptual relationship to the clustering problem as the longest path problem has to the shortest path problem. All students were invited to use interactive visualization environments. Here, students could run visualizations of unlabeled sorting algorithms and test them on different inputs, construct paths from a given source node to a sink node on an undirected, weighted graph, as well as assign classes to randomly generated color-coded coordinates on a plane. We measured how students interacted with the visualization environment by administering surveys that focused on perceptions of induced frustration and curiosity to learn more.

2.2 Analysis Plan

To answer RQ1, we graded constructive reasoning and conceptual understanding outcomes on a 5-point scale, in the former case by identifying meaningful elaborations beyond what we presented in class, and in the latter case by identifying correct answers to our questions. Since our sample sizes were small, we used Bayesian analyses to compare the learning outcomes between our conditions. Specifically, we carried out a Bayesian Mann-Whitney U-test with 1000 samples and computed Bayes Factor BF_{01} to test the null hypothesis that administering FDS would have no effect on conceptual understanding and constructive reasoning. To answer RQ2, we conducted a content analysis of answers to analyze student perceptions of the usefulness of visualization modules. We then used the log data to find the frequency of students' interaction with the modules and calculated correlation between the quality of solutions and the use of the visualization. Interaction events, included, for example, actions such as the addition or removal of nodes in the graph. To answer RQ3, we quantitatively and qualitatively analyzed student responses from relevant surveys [8].

3 Results

3.1 Learning Outcomes (RQ1)

Results for one of the topics (shortest path) showed that students who received FDS scored similarly ($M = 2.5$, $SD = 1.07$) on the conceptual understanding

posttest as students who did not receive FDS ($M = 2.61$, $SD = 1.26$). Although this difference was not statistically significant ($p = 0.32$) and we failed to reject the null hypothesis, this comparison had a BF_{01} of 3.17, indicating positive or substantial odds favoring the null. The effect size for this comparison was small (Cohen's d -0.095). For constructive reasoning, students receiving FDS scored descriptively higher ($M = 1.55$, $SD = 1.65$) relative to students not receiving FDS ($M = 1.21$, $SD = 1.87$), with a moderate effect size (Cohen's d 0.194), despite non-significance of results ($p = 0.39$). This comparison had a BF_{01} of 2.55, indicating only weak or anecdotal odds favoring the null. Taken together, these results partially support hypothesis H1 of students in the FDS condition scoring higher on the conceptual understanding posttest and constructive reasoning, however only for the latter.

3.2 Use of the Visualization Module (RQ2)

Of the 64 students who participated in the second PF session on shortest path algorithms, all answered a question about the visualization module in a survey provided immediately after the end of the problem-solving phase. 54 students (84%) responded affirmatively when asked if they found the visualization module useful, 6 (9%) responded with a "no" or "not really," and 4 (6%) had mixed reactions such as "a little." No student reported that they did not use it. Of those who explained why the module was helpful, responses focused on reasons such as it "made me realize that my algorithm is crap," helped to "generate an idea," and "just saves time." Of those who had critical comments, one student claimed that while it was better than nothing, it did not help with developing code as it did not support "call stacks and such." One student said that they preferred to work out the solution on pen and paper.

The responses to the survey for the clustering algorithms topic, which was administered online (and outside of class time) a few days after the lecture, showed that 15/25 (60%) students found the visualization module useful, 5 (20%) did not, 4 (16%) had mixed responses such as "a little bit", and one student (4%) claimed not to have looked at it. Two students left technical suggestions about ways to improve the module. In terms of logged data, we saw a fairly high level of engagement, evidenced by the frequency of interaction events per student (average of 14). The frequency of interactions, however did not correlate with learning outcomes ($\rho = 0.075$), suggesting the need to quantify the quality of student problem-solving actions in future work. Taken together, these results support hypothesis H2 of students perceiving the interactive AV environment to be useful in facilitating their problem-solving (with or without FDS).

3.3 Underlying Affective Factors (RQ3)

Results from student reactions collected indicated that 27/62 (43.5%) students responded affirmatively to whether they wanted to learn more about the topic, 12 had reserved responses such as "kind of" or "a little bit," and 23 responded negatively. Overall, there was no difference in reported curiosity ($M = 0.47$, SD

= 0.51 for FDS versus $M = 0.41$, $SD = 0.5$ for control, $BF_{01} = 3.27$, Cohen's d 0.121). Some students further elaborated on their answers. Of those who responded affirmatively, some wrote that "you get curious because you failed," "I have a lot to learn," and that their failure to solve the exercise prompted them to find and study Dijkstra's algorithm. Of those who had reserved responses, one wrote that they would have preferred to discuss the exercise in small groups. Of other students who submitted negative responses, one wrote that the topic itself was "super interesting" but the exercise itself made them feel frustrated.

We asked students directly about frustration in the same post-experiment survey and also found it to be similar across our conditions ($M = 0.73$, $SD = 0.45$ for FDS versus $M = 0.79$, $SD = 0.41$ for control, $BF_{01} = 3.29$, Cohen's d 0.141). Of the 64 students who responded, 49 reported that the exercise was frustrating to solve. Many of those students added qualifying statements such as "code always frustrates me", "there's always some kind of frustration with coding for me", "on the other hand I am glad that I did find a part of a solution", and "I feel very lost in [the course] in general." Taken together, these results do not support hypothesis H3 of relevant affective factors differentially impacting how students perceive and learn from PF and FDS.

4 Discussion and Conclusion

Our first research question (RQ1) focused on whether students who received additionally received FDS within PF sessions would demonstrate improved learning outcomes. We expected our results to align with prior work in data science education [8, 10]. Despite the non-significance of results (owing to our small sample size of 41 students), evidence from Bayesian analyses suggests that students exposed to FDS had conceptual understanding posttest scores similar to (not worse than) students who did not receive FDS. We further found an effect size ($d = 0.194$) favoring FDS for students' constructive reasoning. A contextual interpretation of this effect size, drawing on empirical research from the highest-quality field research on factors affecting objective educational outcomes [6], suggests that our effects are large and correspond to the effects of having a very high-quality teacher (versus an average teacher) for one year [2]. Simply put, our effect size estimate of $d = 0.194$ translates to a 55.5% chance that a person picked at random from the FDS group will have a higher quality of constructive reasoning than a person picked at random from a control group not receiving FDS.

Our second research question (RQ2) focused on the effects of the interactive visualization module and the extent to which it helped students during the problem-solving phase. The survey responses to this question ($N = 99$ in total) were very positive, suggesting high perceptions of usefulness among students. We could not find direct evidence that using the module improved learning outcomes, but a possible explanation is that high-performing students may not have needed the visualization module as much to explore the problem space.

Our third research question (RQ3) focused on the role that affective factors may play in facilitating learning from FDS in generative problem-solving.

When working with algorithmic representations deliberately designed to lead to failures, we posit that students would naturally experience frustration and discomfort. However, because this discomfort fuels task progress via problem-space exploration in the presence of FDS, students have a chance to explore relevant problem parameters and develop intuition for what (does not) work. With improved awareness of knowledge gaps, students are better poised to be interested in and learn from the canonical solution. Results showed that not only are FDS students similarly curious, but they also do not experience more frustration.

Acknowledgments

We thank Gustav Hammarhjelm and Dr. Tracy Ewen for valuable feedback on an earlier version of the paper and Dr. Ralf Sasse for helping organize the study.

References

1. Caceffo, R., Gama, G., Azevedo, R.: Exploring active learning approaches to computer science classes. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education (2018)
2. Hanushek, E.A.: Valuing teachers: How much is a good teacher worth. Education next **11**(3), 40–45 (2011)
3. Hao, Q., Barnes, B., Wright, E., Kim, E.: Effects of active learning environments and instructional methods in computer science education. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. pp. 934–939 (2018)
4. Hundhausen, C.D., Douglas, S.A., Stasko, J.T.: A meta-study of algorithm visualization effectiveness. Journal of Visual Languages & Computing **13**(3), 259–290 (2002)
5. Kapur, M., Bielaczyc, K.: Designing for productive failure. Journal of the Learning Sciences **21**(1), 45–83 (2012)
6. Kraft, M.A.: Interpreting effect sizes of education interventions. Educational Researcher **49**(4), 241–253 (2020)
7. Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., et al.: Exploring the role of visualization and engagement in computer science education. In: Working group reports from ITiCSE on Innovation and technology in computer science education, pp. 131–152 (2002)
8. Sinha, T., Kapur, M.: Robust effects of the efficacy of explicit failure-driven scaffolding in problem-solving prior to instruction: A replication and extension. Learning and Instruction **75**, 101488 (2021)
9. Sinha, T., Kapur, M.: When problem solving followed by instruction works: Evidence for productive failure. Review of Educational Research **91**(5), 761–798 (2021)
10. Sinha, T., Kapur, M., West, R., Catasta, M., Hauswirth, M., Trninic, D.: Differential benefits of explicit failure-driven and success-driven scaffolding in problem-solving prior to instruction. Journal of Educational Psychology **113**(3), 530 (2021)
11. Thorgeirsson, S., Su, Z.: Algot: An educational programming language with human-intuitive visual syntax. In: 2021 IEEE Symposium on Visual Languages and Human-Centric Computing. pp. 1–5. IEEE (2021)