Pilot Study on Optimal Task Scheduling in Learning

Lin Ling
City University of Hong Kong
83 Tat Chee Ave, Kowloon Tong, Hong Kong
linling2-c@my.cityu.edu.hk

Chee Wei Tan
City University of Hong Kong
83 Tat Chee Ave, Kowloon Tong, Hong Kong
cheewtan@cityu.edu.hk

ABSTRACT
Living in an information era where various online learning contents are rapidly available, students often learn with a combination of multiple learning tasks. In this work we explore the possibilities of using optimization theory to find the optimal trade-off between the time invested in two different completing learning tasks for each individual student. We show that the problem can be formulated as a linear programming problem, which can be efficiently solved to determine the optimal amount of time for each task. We also report our ongoing attempts to apply this theory to our Facebook Messenger chatbot software that can optimize the trade-off between learning and self-assessing in form of MCQs on the chatbot platform.

ACM Classification Keywords
G.1.6 Numerical Analysis: Optimization; K.3.1 Computers and Education: Computer Uses in Education

Author Keywords
Optimization Theory; Personalized Learning; Learning Effectiveness; Learning Task Scheduling; Chatbot Software.

INTRODUCTION
Online learning that makes use of mobile computing devices and software technologies to connect learners and educators has become increasingly popular in recent years. This has engendered a trend of personalized learning that serves to enhance individual learning experience and even make academic acceleration possible. Personalized learning overcomes the scale-efficacy trade-off of learning by adjusting the course content to be different for each student. The idea of leveraging technologies to shape the learning curves of individual students for certain subjects such as mathematics is appealing as progressively harder topics can be served and assessed automatically while tracking learning performance efficiently.

Learning with computers dates back to the 1960's work by C. L. Liu in [2] where he proposed a computer-based method to automate teaching linear algebra by assigning students to different groups with each group having different teaching strategies to accommodate the learning progress of individual students. Later, S. Papert, one of the pioneers of artificial intelligence, also proposed using computers to improve mathematics education [5]. Apart from offering personalized learning content, computer-based methods provide students with instant feedback. In his Turing lecture in 1969, M. Minsky advocated that computer science educators should help students to "debug" their thinking process, so that when facing difficulties, students would think "How can I make myself better at this?" and possibly be assisted by computational procedures [4]. In [7], J. Ullman proposed the Gradiance On-Line Accelerated Learning (GOAL) as a means of automating self-assessment quizzes that center around a root-question design to vary difficulty levels. Clearly, computers that generate self-paced content, instant assessment feedback and predictive analytics have a definite role to help students to "debug" their thinking processes and to succeed in learning.

Even though there are numerous work in online knowledge delivery, our work is unique in the sense that it is strongly motivated by recognizing that learning a subject often invariably means alternating between two complementary sub-processes. For example, learning might be in a passively instructional mode or actively hands-on mode. Learning might be assimilating standard-abiding curriculum content or enrichment-type content (e.g., educational games, mathematics gamification [6]). Another example is *flipped classroom* composing of learning inside and outside classroom through digital video recording. How to coordinate this alternating process by controlling the amount of time for these two tasks? When intermediate milestones are imposed, how to introduce increasingly challenging tasks while adapting to their individual learning curves? In answering these questions, we will highlight the role of computing technologies and predictive analytic software to scale up, thereby optimizing the delivery of alternating tasks to shape individual learning behavior and performance.

A similar problem was brought out by S. Boyd as an exercise question in his book [1], which discussed an optimal way for students to split their time between learning theories in a field and applying the skills/theories that they learned. S. Low also discussed a similar topic in one of his blog posts to find the optimal trade-off between learning and producing [3]. In this paper we develop this idea and put it into software for addressing personalized learning scenario, and to find an optimal scheduling between different learning tasks.

OPTIMAL TASK SCHEDULING FRAMEWORK
In this section, we formulate the problem of task scheduling for personalized learning as a mathematical optimization problem.

L@S 2018, June 26–28, 2018, London, United Kingdom

ISBN 978-1-4503-5886-6/18/06…$15.00

DOI: https://doi.org/10.1145/3231644.3231677

In this particular example, we are scheduling the time invested in the following two tasks:

1. In-class learning (e.g., from a human teacher)
2. After-class personalized self-assessment

Note that although we are focusing on the two tasks mentioned above, the same algorithm can be applied to a variety of task scheduling problems in education and learning.

Optimization Problem Formulation

To model the problem, we assume that there are n "learning units" for each student. A learning unit is the individual time unit to be split by the two tasks. For example, for a one semester course, a learning unit could be a week's time, and there might be $n = 13$ learning units for each student in one semester.

Our goal is to find a vector $x \in \mathbb{R}^n$, such that it maximizes the learning outcome of the student. Here the entries x_i represents the percentage of time in each learning unit to be devoted to in-class learning. Thus, we also require that $0 \preceq x \preceq 1$, where $x_i = 1$ indicates all the time in learning unit i is invested in in-class learning, and $x_i = 0$ means the opposite.

To formulate the problem as a mathematical optimization problem, we also need to find a way to characterize the learning outcome of each student. For assessing the learning outcome, we can simply test the students with quizzes and exams, but modelling the learning outcome as a function to x is nontrivial in general. Here we introduce three personal parameters for each student: α, β and θ, where $\alpha \in (0,1)$ and $\beta \in (0,1)$ measures the learning effectiveness for in-class learning and after-class learning respectively, and $\theta \in (0,1)$ is a measurement of the speed of learning and adapting new knowledge. All of the three parameters take different values for different students.

With these parameters we can model the learning effectiveness over all n learning units, denoted as e, where

$$e_{i+1} = (1-\theta)e_i + \theta(\alpha x_{i+1} + \beta(1-x_{i+1})).$$

Intuitively, e is an accumulated value computed as the weighted sum of the current e and the total learning effectiveness for the current learning unit. The rate of change of e depends on the θ value. The larger θ is, the faster the student learns, and the more sensitive the accumulated effectiveness is to α and β values.

To evaluate the overall learning outcome of a student, we need to compute the accumulated learning effectiveness till the last learning unit (denoted as E). Assuming $e_0 = 0$, we have:

$$E = e_n = \sum_{i=1}^{n} \theta(1-\theta)^{n-i}((\alpha-\beta)x_i + \beta).$$

Let $t_i = \theta(1-\theta)^{n-i}$ and $t = [t_1 \quad t_2 \quad \cdots \quad t_n]^T$, we can rewrite the above equation in a compact vector form:

$$E = (\alpha-\beta)t^T x + \beta t^T \mathbf{1}.$$

Now we can express our problem as (dropping constant terms):

$$\begin{aligned} \text{minimize} \quad & (\beta-\alpha)t^T x \\ \text{subject to} \quad & 0 \preceq x \preceq 1. \end{aligned} \tag{1}$$

Optimality Characterization

The Problem (1) is a simple linear programming problem, and we can easily see by inspection that the optimal x depends on the relative value of α and β. More specifically:

$$x^* = \begin{cases} \mathbf{1}, & \text{if } \alpha \geq \beta \\ \mathbf{0}, & \text{otherwise} \end{cases}$$

This is just saying that, for a specific student, if in-class learning is more effective, we devote all the time in every single learning unit to in-class learning. Otherwise, all the time will be devoted to after-class personalized learning.

This is a simple yet trivial result. It makes sense but is not practical in reality, since we want the personalized learning methods to be used as assistance to the traditional in-class learning, instead of replacing it. Also, from a practical perspective, the personalized learning methods cannot be used until the student has learned an adequate amount of basic knowledge in that field.

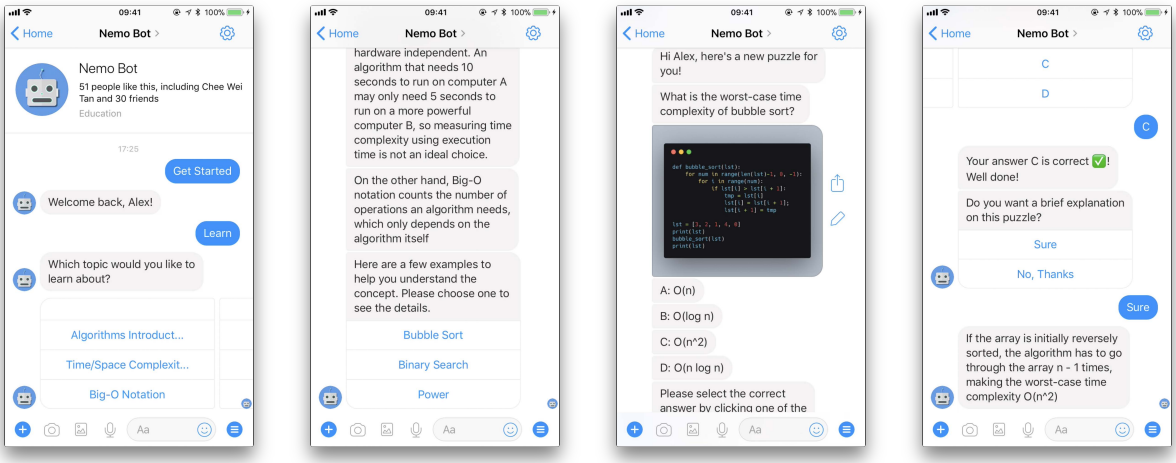
Thus, to improve the modeling, another set of constraints is needed in our optimization problem to ensure that students have learned enough basic knowledge before attempting the personalized self-assessments. Here we use I_k and A_k to denote the total number of learning units a student has invested in in-class learning and after-class personalized learning, respectively, up to a given learning unit k. That is:

$$\begin{aligned} I_k &= \sum_{i=1}^{k} x_i \\ A_k &= \sum_{i=1}^{k} (1-x_i) = k - I_k. \end{aligned}$$

We want A_k to be constrained by I_k, so we need a function f_c to construct the constraint $A_k \leq f_c(I_k)$. The revised problem can be expressed as (ignoring constant terms):

$$\begin{aligned} \text{minimize} \quad & (\beta-\alpha)t^T x \\ \text{subject to} \quad & A_k \leq f_c(I_k) \ \ \forall k \in \{1, \cdots, n\}, \\ & 0 \preceq x \preceq 1. \end{aligned} \tag{2}$$

Note that here we require f_c to be a monotonic increasing function, as in reality the amount of after-class self-learning material accessible should strictly increase as the student spends more time for in-class learning.



(a) (b) (c) (d)

Figure 1: Nemo Bot software interface. (a) and (b) illustrate how the bot can provide factual learning with guidance; (c) and (d) illustrate how students can learn by answering a series of MCQs.

By expanding the A_k and I_k and rearranging the terms in problem (2), we can reformulate it into the following equivalent problem:

$$\begin{aligned} \text{minimize} \quad & (\beta - \alpha) t^T x \\ \text{subject to} \quad & \sum_{i=1}^{k} x_i \geq g_c^{-1}(k) \quad \forall k \in \{1, \cdots, n\}, \\ & 0 \preceq x \preceq 1, \end{aligned} \tag{3}$$

where $g_c(k) = f_c(k) + k$, and g_c^{-1} is the inverse function of g_c. The reformulated problem (3) is a linear programming problem (LP), which is a well-studied area of mathematical optimization, and can be solved efficiently using various solvers and algorithms.

APPLICATION IN MESSENGER CHATBOT

We are also interested to apply the optimal learning task scheduling theory discussed above in a real-world scenario and to observe its learning efficacy. Specifically, we are building a Messenger chatbot called Nemo Bot (available at https://m.me/454163798317367), where the theory is incorporated and offers students a way of learning different topics in Computer Science/Mathematics at their own pace. Students can learn a selected topic through a combination of two methods: by interacting with the bot where hand-crafted examples and video resources will be given, as well as by answering carefully-designed MCQs that are closely related to the said topic. This is shown in Figure 1, and a high-level overview of the Nemo Bot system is shown in Figure 2.

We chose MCQ as the form of assessment on Nemo Bot because it is easy for the system to automatically grade the answers from students, and therefore providing instant feedback as well as hints when they get a wrong answer. This was largely inspired by the work of J. Ullman on the Gradiance

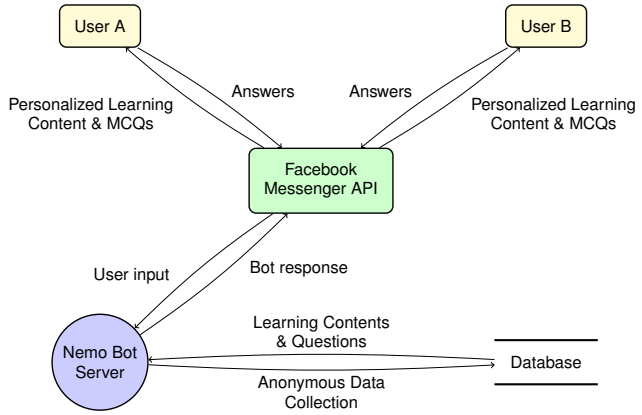


Figure 2: High-level overview of the Nemo Bot system. The learning content distribution and user input/answer are handled by the Facebook Messenger API. Our server communicate with the Messenger API and store anonymous user data to the database for future analysis.

On-Line Accelerated Learning (GOAL) framework, in which he described a method of constructing a set of MCQs from several "root questions" and feeding them to the students to replace the traditional homework [7].

In traditional homework, the answer sheets have to be graded by the teachers and returned to the students weeks later, and there is a large chance that the students may not revisit and rework the problems they have missed, and in some cases the answer sheets might not even be returned back to the students. On the contrary, in the GOAL framework, students get instant feedback and hints, but they are also encouraged to work on the problems repeatedly until they get the correct answers. In our Nemo Bot system, we further improve this by showing the related content to students again if they make a mistake

in the MCQs, so they can revisit the content before directly reattempting the problems, which helps ensure that they truly understand the topic when they finish the MCQs.

In our Nemo Bot case, we are optimizing the task scheduling between learning and answering MCQs, instead of between in-class learning and after-class learning, due to the fact that it is impractical for us to manually control the learning time of students in the classroom. Similar to the in/after class case, students need to learn enough basic knowledge from the bot before it is meaningful for them to attempt the MCQs. Thus our discussion on the f_c constraints also applies here.

To put our optimal learning task scheduling system into test, we will release it to the participants of the Computer Science Challenge 2018 (https://cschallenge.cs.cityu.edu.hk/) organized by City University of Hong Kong, where students will compete in small teams in fun games while gaining insight and intuition in various topics in Mathematics and Computer Science. One month prior to the actual event, we will invite the students to learn a mini course with Nemo Bot on basic C programming and data structures, which is closely related to the Task 3 in the Compute Science Challenge. We hope to observe a correlation between the task score of students and their level of engagement in learning with Nemo Bot.

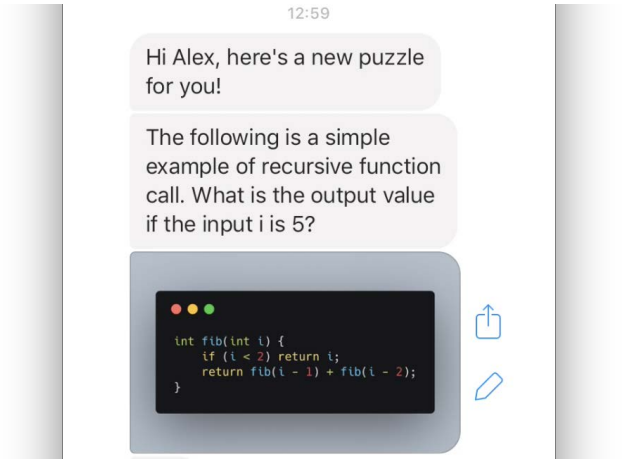


Figure 3: An example MCQ about recursive function on the Nemo Bot platform

FUTURE WORK

In reality, learning activities rarely happen with only two learning tasks. A student might have traditional in-class learning as well as multiple forms of after-class self-learning or self-assessment. The current optimal learning task scheduling system can only handle two competing learning tasks, and therefore we need to extend our theoretical framework to produce the optimal scheduling between three or more tasks.

Another problem with the current work is that the optimal task scheduling it produces often consists of successive learning units completely devoted to a learning method. For example, in the in-class learning and after-class personalized learning example in a 13 weeks course, the system might suggest a student to devote all her studies time to in-class learning in the first 7 weeks, and then devoted to after-class learning in the remaining 6 weeks. This may not be feasible, as in practice students can quickly get bored and therefore become less effective learners if they are forced to do a single learning task for a relatively long period.

To incorporate this into our system, we need to introduce some kind of mechanisms that penalize unduly long successive learning units in a single learning task. We can do this by progressively reducing the student's learning effectiveness parameters (α and β in the above case) value if the number of successive learning units for a learning task exceeds certain threshold.

CONCLUSION

In this work, we presented our optimal learning task scheduling framework, a system that uses mathematical optimization to find the most effective trade-off between the time invested in multiple learning tasks, which is a problem that often arise in the era of personalized learning. We inspected the mathematical structure of such problem and formulated a linear programming problem which can be efficiently solved. We also reported our ongoing efforts of applying the theory in a Facebook Messenger learning chatbot software to verify the theory in real-world scenarios.

REFERENCES

1. Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge University Press.
2. Chung Laung. Liu. 1960. A study in machine-aided learning. *Massachusetts Institute of Technology, M.S. Thesis* (1960).
3. Steven Low. 2015. Another puzzle: produce or learn? (2015). https://rigorandrelevance.wordpress.com/2015/01/10/another-puzzle-produce-or-learn/
4. Marvin Minsky. 1970. Form and Content in Computer Science (1970 ACM Turing Lecture). *J. ACM* 17, 2 (April 1970), 197–215. DOI: http://dx.doi.org/10.1145/321574.321575
5. Seymour A Papert. 1973. Uses of technology to enhance education. (1973).
6. Chee-wei Tan, Pei-duo Yu, and Lin Ling. 2018. Teaching Computational Thinking by Gamification of K - 12 Mathematics: Mobile App Math Games in Mathematics and Computer Science Tournament. In *Computational Thinking Education*, Hal Abelson and Siu Cheung Kong (Eds.). Springer, Hong Kong.
7. Jeffrey D. Ullman. 2005. Gradiance On-Line Accelerated Learning. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38 (ACSC '05)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 3–6. http://dl.acm.org/citation.cfm?id=1082161.1082162