Integrating Learning by Explanation, Abstraction, Analogical Reasoning, Experimentation, Observation, Representation Change, Meta-Reasoning, and Static Analysis in Prodigy to Solve the Advisor Scheduling Problem

Abstract
The problem of advisor scheduling has perplexed students for many years. Advisors are a scarce resource that must be managed carefully. In this paper, we prove that advisor scheduling is NP-hard by reduction from the PSPACE-complete problem of thesis defense scheduling. We then present an innovative solution to this problem that employs massively integrated learning, which combines every learning approach that we could think of. Our experiments show that this novel learning method provides a ten-fold improvement in the rate of successfully executed advisor meetings.

1 Introduction
Advisor scheduling is the problem of scheduling and executing a meeting with an academic advisor. While it is often relatively easy to schedule a meeting, it is not successful unless the advisor actually shows up and participates in the meeting. Advisors tend to be very busy and are often involved in multiple meetings with multiple students, colleagues, and DARPA Program Managers. And real-world events often conspire against the successful execution of a meeting.

Given these challenges, students may have to go to great lengths to obtain a meeting. For instance, one graduate student in Western Pennsylvania reportedly wrote a program that monitored his Ph.D. advisor’s office computer on weekends, and notified him as soon as his advisor began using the keyboard so that he could be the first to knock on his door (Anonymous, 1990). Clearly, a successful advisor scheduling system has to be both intelligent and clever to develop strategies for outmaneuvering the advisor. Although it is not necessarily an adversarial relationship, the problem does have elements of multi-player games.

This paper builds on the Prodigy architecture, as described by Minton, Carbonell, Knoblock, Kuokka, Etzioni and Gil (1989):

Prodigy is a domain-independent problem solver that acquires new knowledge by analyzing its experiences and interacting with an expert. In essence, the Prodigy architecture is inspired by observation of how intelligent human students transition gradually from novice to increasingly more expert performance by being taught through problem-solving practice and much trial and error. Although not a fine-grain cognitive model, Prodigy nevertheless strives to retain the human flexibility of applying focused expertise when available, and relying on less focused general problem-solving behavior when domain knowledge fails to produce an adequate answer. Thus, the learning components discussed below acquire either factual domain knowledge or domain-specific control knowledge, both necessary components for search-limited, knowledge-intensive, expert behavior. This new knowledge
is then used by the problem solver (whose structure does not change) when encountering new, progressively more difficult problems in the same domain.

In this paper, we build on the work of Minton (1988), Knoblock (1991), Kuokka (1990), Etzioni (1993), Gil (1992), Veloso (1992), Perez (1995), Wang (1996), Blythe (1998), and Fink (1999), combining their methods to create a new super duper learning approach far superior to any single one of these methods. To combine the individual methods, we employ a new method, building on Wolpert’s stacking approach (Wolpert, 1992), which was motivated by Prodigy’s excellent performance in the blocks world.

In the following sections we state our theoretical contribution, outline the approach, describe our real-world evaluation in detail, and discuss the far-reaching implications of this work.

2 Theoretical Result

We prove that advisor scheduling is NP-hard by reducing the problem of defense scheduling, which is known to be PSPACE-complete. We ignore the additional twist of finding an appropriate room for the defense, which is known to make the problem even harder and has caused some students to spend extra years in school, if not quit Ph.D. programs altogether.

Theorem Advisor scheduling is NP-hard.

Proof: the problem of defense scheduling requires multiple advisors to be in the same place, and time, for a lengthy defense whose outcome is a foregone conclusion. Moreover, at least one advisor has to be awake at any given point in time. Thus, you can only schedule a defense if you have become highly adept at advisor scheduling. In fact, some have argued that a solution to the advisor scheduling problem is the key requirement for graduation because once your advisor is sufficiently tired of meeting with you he will insist that you graduate as a way of clearing his calendar. □

3 Technical Approach

Stacking is an ensemble-learning method, which in previous work has been used to combine multiple learners. However, previous researchers have employed stacking in a very limited way, focusing on just a few methods. In this paper, we introduce the idea of massively-integrated learning, which we refer to as wide learning. In wide learning, a problem solver first tries one learning method. If the problem is solved, the system then selects another problem to solve. If the system cannot solve the original problem, and it is still functioning, it then attempts to employ a second learning method, and so on.

Our approach integrates Learning by Explanation, Abstraction, Analogical Reasoning, Experimentation, Observation, Representation-Change, Meta-Reasoning, and Static Analysis. We call the resulting, amalgamated system “Prodigy+++. In the remainder of this section, we consider an illustrative scenario, and describe how Prodigy+++ successfully solves an instance of the advisor-scheduling problem.

Let’s consider a motivating example. We suppose that in the initial state Prodigy+++ has an advisor whose office is Gates Hillman Center 6721. The system is given the goal of scheduling a meeting with its advisor. We use a robot supplied by Veloso Robotics, Inc. to actually test the execution of the system. We assume that the advisor is in an unknown state and part of Prodigy+++’s mission is to ascertain the current state in order to successfully schedule and execute the meeting.

In our scenario, Prodigy+++ begins by invoking Gil’s learning by experimentation module to create an operator to schedule a meeting. Since the operators in this domain are initially unknown, it experiments on other faculty members and requests a meeting with Robert Frederking. It successfully schedules and executes the meeting and Prodigy+++ is now confident that it can solve the advisor scheduling task. So Prodigy+++ applies the new operator, but the advisor is not in the office and the plan fails. The system then applies the work of Blythe on planning under uncertainty in dynamic domains to produce a plan that will result in putting the advisor in a known state. In this case, Prodigy+++ pulls the fire alarm so the advisor’s location is now known - i.e., outside the building. After encountering significant weather issues, the explanation-based learning module learns from failure and determines that an alternative approach is necessary. Prodigy+++ next invokes Veloso’s analogical reasoning module, which hypothesizes that

1 We removed all soccer-playing paraphernalia including shin guards and cleats prior to the evaluation.
scheduling a meeting with one’s advisor is a lot like scheduling a date. Despite a liberal douse of perfume and fashionable dress, the advisor still does not show up. After trying a massive number of learning methods, the system finally succeeds through its highly advanced abstraction learning method. The system drops predicates having to do with location and time and as a result it executes a plan to stand still waiting in the hallway indefinitely. During this time period, the advisor passes by and Prodigy+++ declares success.

4 Evaluation

In order to evaluate the system, we compared Prodigy+++ to both the original Prodigy system and to randomly selected students. We asked each to schedule and execute multiple meetings with a given advisor and compared the results. The results demonstrate the challenging nature of the problem and the power of wide learning. As shown in Figure 1, base-level Prodigy was unsuccessful at scheduling any meetings, although the system did build an impressive staircase in an attempt to find the advisor.

![Graph showing performance improvement](image)

**Figure 1:** Experimental comparison showing 10-fold improvement in advisor scheduling

The human students did occasionally schedule and execute meetings, however their success rate was only 0.001% (i.e., one successful meeting for every 1,000 attempts). As shown in Figure 1, Prodigy+++ achieved a 10-fold increase in its success rate by obtaining a success rate of 0.01%. We note that the length of the meetings was rather short since the advisor simply passed by in the hallway. (In future work, we intend to address the issue of plan quality.)

5 Related Work

Previous work on problem solving has combined at most two learning methods. For example Knoblock, Minton, and Etzioni (1991) combined abstraction and explanation-based learning in Prodigy. Although this approach performed awesomely in Towers of Hanoi and block stacking, advisor scheduling is a much more challenging problem. There is no amount of block stacking that will ever arrange a meeting with your advisor. The SOAR architecture (Laird, Newell, and Rosenbloom, 1987) also employed multiple learning methods; however, the authors acknowledge that these are weak methods. Consequently, SOAR could never solve such a challenging problem.

6 Conclusion

This paper has three important contributions. First, we proved that advisor scheduling is not just hard, but really hard. Second, we presented a solution to the advisor-scheduling problem, which has important ramifications for students worldwide. Finally, we presented a general, architectural advance: if the combination of two learning methods is good, then many more is even better. We believe that massively integrated learning, which we refer to as wide learning, has the potential to have a significantly greater impact than deep learning. In particular, wide learning can exploit a greater surface area than deep learning, which is limited by depth considerations.

In future work we hope to return to block stacking and show that the techniques presented here generalize to both challenging and trivial problems. We also plan to combine wide and deep learning to produce a method that is both wide and deep.

Acknowledgements

This research was funded by a three-letter agency that starts with the letter “N” that wishes to remain anonymous. The views and conclusion contained in this paper are not those of the authors and only represent the official opinion and policy of the anonymous agency as well as all of the people and agencies connected with them. Any errors are due to flawed graduate training of the authors, and should be pointed out immediately to Jaime Carbonell, the responsible party.
References


