What can Automated Planning do for Intelligent Tutoring Systems?

Sachin Grover[*] and **Tathagata Chakraborti**[*] and **Subbarao Kambhampati**
Arizona State University, Tempe, AZ 85281 USA
{ sgrover6, tchakra2, rao } @ asu.edu

Abstract

In this paper, we build on the latest in automated planning techniques to develop a generalized framework for course-independent design of Intelligent Tutoring Systems (ITSs). This is meant to provide targeted and personalized assistance to students, in order to meet the demands of the increasing class size, as well as help instructors who can use higher level specifications to design courses without having to worry about building the course-specific tutoring assistance. Thus the aim of this paper is to demonstrate what automated planning can bring to the table for the design of course-independent ITS features. We will illustrate these capabilities in **`Dragoon`**, an ITS deployed at Arizona State University.

1 Introduction

While the last decade has seen massive advances in technologies aimed at creation and dissemination of knowledge across a variety of platforms, concerns remain as to how effectively this knowledge is absorbed at the user (student) end. This is especially true for both massive open online courses (MOOCs) and also for (rapidly growing sizes of) physical classrooms where targeted attention towards individual students is often hard to provide. The state-of-the-art in student and instructor support technology has traditionally struggled to catch up with the demands of the rapidly evolving landscape of education in the 21^{st} century. In this paper, we address this by proposing a framework for the design of generic course-independent student and instructor support capabilities using techniques in the field of human-aware planning, and demonstrate those features in **`Dragoon`**, a celebrated intelligent tutoring system.

1.1 Learning 2.0

The world of learning is indeed changing fast - information can now be provided across a variety of platforms to large groups of people who can access *on demand* knowledge and participate in the learning process as a *community*. This is the Learning 2.0 paradigm (Seely Brown and Adler 2008), and requires a rethink of the affordances (McLoughlin and Lee 2007) expected from current learning tools.

Learning on Demand Learning on demand refers to the increasing popularity of individual student-centric and topic-driven learning achieved on the web – i.e. students pick a particular topic they want to learn about and actively consume content just based on that, instead of participating in an entire class or following through an entire curriculum. For example, consider that you want to learn about regression – you could log on to Coursera, complete the relevant tutorials and assignments on regression, and leave the course. This requires a rethink of traditional curriculum generation and course recommendation approaches that would traditionally compute end to end curricula for an entire class. It follows that such new approaches must be able to leverage detailed student models to provide effective support.

Social Learning One of the many advantages of social platforms for learning is peer feedback and community participation – i.e. social learning (Burke 2011). This involves two critical aspects – *knowledge advancement as a community* (Scardamalia and Bereiter 2006) and *information processing* (Webb 2013) on the part of the individual student as a member of that community. In a sense, this can even be seen as a proxy towards providing individual classroom attention from the instructor. However, forming study partners remains an arduous task, especially in large classrooms such as in online learning communities where students usually do not know most of their classmates (or their skill sets). It is also fraught with the usual pitfalls associated with group work including individual students hogging all the group activity or slackers not contributing to the group activity at all (Mesch 1991). Without principled drivers for building in-class communities that can promote learning, effective collaborations are hard to achieve. As such, forming useful teams for collaborative study can become a problem by itself rather than a facilitator for learning to the extent that students can end up spending too much effort in forming and maintaining teams or just prefer to study by themselves, thus leaving the potential benefits of a social learning environment largely untapped. Recent work has shown that peer recommendations can have positive impact (Labarthe et al. 2016) on student engagement but has remained ambiguous (Bouchet et al. 2017) as to the best way to go about it.

1.2 A Brief History of ITS and AI

ITSs are aimed to provide personalized support to students and bring in expert (human) tutors in the loop wherever necessary, thus reducing the burden on the instructor as well as improving the learning experience of the student. In fact, it has been shown that when designed correctly, an ITS can

[*]Authors marked with [*] contributed equally.

be as effective as a human teacher (VanLehn 2011). A thorough description of the different components of ITSs can be found in (Vanlehn 2006). Existing applications of such systems range from solving numerical problems like Andes (Gertner and VanLehn 2000) which can help in teaching basic laws of physics (Schulze et al. 2000), Dragoon (VanLehn et al. 2017), Q&A type problems as in Autotutor (Graesser et al. 2005) or for an SQL tutor (Mitrovic 2003). ITSs, of course, go beyond individual information processing stage and find uses in knowledge building as a community (Magnisalis, Demetriadis, and Karakostas 2011) as well, thereby embracing the principles of the Learning 2.0 paradigm.

Student Assessment Models One of the most important capabilities an ITS needs to have is to be able to estimate the (mental) model or capabilities of the student. This has been explored in the context of the (1) *item response theory (IRT)* (Hambleton, Swaminathan, and Rogers 1991) which treats learning and testing as separate processes and the (2) *Bayesian knowledge tracing (BKT) theory* (Corbett and Anderson 1994) which considers a more dynamic model of the student state. The latter becomes more relevant in the context of ITSs that can provide more dynamic feedback and hints as discussed next. Indeed this is an issue where AI techniques have been deployed before for dynamic modeling of the evolution of the student model in terms of knowledge components, concentration / focus levels, etc. (Murray, VanLehn, and Mostow 2004). This includes different techniques such as *decision theoretic approaches* (i.e. Markov Decision Processes or MDPs) (Murray, VanLehn, and Mostow 2004; Murray and VanLehn 2006), and *reinforcement learning* (Chi, VanLehn, and Litman 2010; Mandel et al. 2014; Mandel 2017). This paper assumes for the most part[1] that these techniques are available and builds on top of that assumption, i.e. being able to estimate the student model is necessary for ITS techniques and we want to demonstrate, from the perspective of automated planning how this can be exploited to provide a better learning experience to a student.

Feedbacks and Hints Once the ITS has estimated a model of the student, it can provide targeted feedback to improve the learning process. Existing work in this area (Barnes and Stamper 2010; Stamper et al. 2013; Rivers and Koedinger 2013; 2017) has largely focused on ITSs operating as *recommender systems*. This paper is largely situated in this space but aimed at providing much more sophisticated feedback in both the inner and outer loops (Vanlehn 2006) of an ITS which requires longer-term sequential reasoning.

1.3 What can planning bring to the table?

Automated planning, as a field, has been around ever since the inception of AI, and is considered a necessary ability of any autonomous system – the ability to reason about and decide on a course of action (CoA) or *plan* given the current state of the world. Many of the challenges faced in the design of an ITS bears parallels to the planning agenda – making a curriculum, solving a given problem, or in general dealing with the combinatorics of orchestrating a class can be potentially seen through the lens of planning, i.e. computing a sequence of steps given a set of constraints. This was the starting point of our investigation in this direction.

However, when operating with humans in the loop, traditional planning techniques are not sufficient (Kambhampati and Talamadupula 2015). A "human-aware" planner must be able to take into account the (mental) model (Chakraborti et al. 2017a) of the user. Recent work (Sengupta et al. 2017) has looked at how planning techniques can evolve in the context of *decision support* to guide the planning process of a *human* decision-maker. This includes support for plan validation, critiquing, recommendation, explanations, and so on. Much of the discussion here derives inspiration from recent advances in the planning community along these directions.

Contributions Thus, to answer the question what automated planning can do for the ITS scene, we build on the following two features of planning techniques –

- *Domain independence* – Planning techniques have been particularly geared towards domain-independent solutions – i.e. algorithms that can work across a variety of domains provided in higher-level specification. This is especially useful in the contexts of ITSs which have traditionally been restricted to class or course specific solutions that do not generalize; and
- *Model-based reasoning* – Personalized support for students require higher level and sequential reasoning about the course and *student models*, planning techniques remain ideally suited for this.

In this paper, we expound on the above two themes to –

- Provide targeted feedback when students are stuck on problems by leveraging the student model; (Section 3.2)
- Compute *on demand* curriculum based on class materials requested by the student; (Section 3.3)
  - We will show how this technique can be used to teach concepts to a student to attain different levels of expertise as desired by the student; and
  - We will show how student models may be composed to form joint plans of study.
- Generate class curriculum in the spirit of social learning by including fellow classmates in a student's curriculum while also guaranteeing desired properties of the curriculum – e.g. that students not only learn but also apply all the concepts at least once. (Section 3.4)

We do not, of course, set out to model the full scope of challenges[2] in building and end-to-end ITS. However, we recognize that much of the existing work on deploying ITS

[1]In fact, the "model reconciliation" technique discussed later can handle uncertain models (Sreedharan, Chakraborti, and Kambhampati 2018) and can even be modified to function as an estimator for the student model but this is outside the scope of the paper.

[2]For example, the current discussion only focuses on the learning and interaction phase and does not include post-hoc reflection / evaluations as explored in (Katz, O'Donnell, and Kay 2000; Katz, Allbritton, and Connelly 2003; Connelly and Katz 2009)

systems, if not in conceptualizing them, has focused on specific learning platforms or courses without any coherent approach or general principles of design and implementation of the roles usually attributed to ITSs. The aim of this paper is thus to introduce techniques from the planning community that can formalize some of these concepts and provide a generalized framework for building such systems from the ground up. This has useful implications for both the planning as well as the educational technologies communities – i.e. the former can provide solutions to existing problems in ITSs (as we demonstrate in this paper) while feedback form the learning community can provide useful feedback towards the refinement of said techniques, including defining new areas of research of mutual interest. The biggest advantage of such an approach, as mentioned above, is that the techniques are domain-independent, i.e. they are defined at the procedural level and can be grounded with the description of a particular course as specified by the instructor. Of course, the problem of knowledge representation is (for a specific course and assignments in it) remain a challenge, but the ITS features themselves generalize given the proposed planning framework.

2 Background

In the following, we will introduce concepts from the planning literature that will be used in the rest of the paper.

A Classical Planning Problem (CPP) (Russell and Norvig 2003) is the tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, and A is a set of actions - and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$. Action $a \in A$ is a tuple $\langle c_a, \mathit{pre}(a), \mathit{eff}^{\pm}(a) \rangle$ where c_a is the cost, and $\mathit{pre}(a), \mathit{eff}^{\pm}(a) \subseteq F$ are the preconditions and add/delete effects, i.e. $\delta_{\mathcal{M}}(s, a) \models \bot$ *if* $s \not\models \mathit{pre}(a)$; *else* $\delta_{\mathcal{M}}(s, a) \models s \cup \mathit{eff}^{+}(a) \setminus \mathit{eff}^{-}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. The cumulative transition function is $\delta_{\mathcal{M}}(s, \langle a_1, a_2, \ldots, a_n \rangle) = \delta_{\mathcal{M}}(\delta_{\mathcal{M}}(s, a_1), \langle a_2, \ldots, a_n \rangle)$.

A CPP is represented using the Planning Domain Definition Language or PDDL (McDermott et al. 1998).

A Plan Generator Module (PGM) (Helmert 2006) computes a solution to a CPP $\mathcal{M}$ as sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. The cost of π is $C(\pi, \mathcal{M}) = \sum_{a \in \pi} c_a$ if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; ∞ otherwise. The cheapest plan $\pi^* = \arg\min_{\pi} C(\pi, \mathcal{M})$ is the optimal plan with cost $C^*_{\mathcal{M}}$.

A Plan Validation Module (PVM) (Howey, Long, and Fox 2004) outputs, given plan π and planning problem $\mathcal{M}$, `True` iff $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; `False` otherwise.

A Plan Recognition Module (PRM) (Ramírez and Geffner 2010) outputs, given a *partial plan* $\hat{\pi}$ and a planning problem $\mathcal{M}$, a plan π that maximizes the probability that $\hat{\pi}$ is a sub-plan of π –

$$\pi \leftarrow \arg\min_{\pi} \mathcal{P}([\hat{\pi}]_{k=0}^{k \leq |\pi|})$$

Note that the above approach does not directly compute this. Instead, we use the compilation approach from (Ramírez and Geffner 2009) to compute *the optimal plan that satisfies a set of observations given a goal* as the output of the PRM.

A Landmark Generation Module (LGM) (Hoffmann, Porteous, and Sebastia 2004) outputs, given a planning problem $\mathcal{M}$, a set of state (or action) landmarks $\mathcal{L}$ containing states (or actions) that must be passed through (or executed) in any satisficing solution of $\mathcal{M}$. Thus –

- An action landmark $a \in A$ requires that $a \in \pi$ $\forall \pi : \delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$.
- A state landmark $s \subseteq F$ is such that $\forall \pi : \delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$, $\exists [\hat{\pi}]_{k=0}^{k \leq |\pi|} : \delta_{\mathcal{M}}(\mathcal{I}, \hat{\pi}) \models s$. (Zhu and Givan 2003)

A Human-Aware Planning Problem (HAP) is given by the tuple $\Psi = \langle \mathcal{M}, \mathcal{M}^H \rangle$ where $\mathcal{M}^H = \langle D^H, \mathcal{I}^H, \mathcal{G}^H \rangle$ is the human's understanding of the planning problem $\mathcal{M}$ (Chakraborti et al. 2017a).

An Explicable Planning Module (EPM) computes a plan π such that it is a satisficing solution to $\mathcal{M}$ and is as close as possible to the expected plan in the human's model (Zhang et al. 2017; 2016; Kulkarni et al. 2016) –

$$C(\pi, \mathcal{M}) \approx C^*_{\mathcal{M}^H}$$

A Plan Explanation Module (PEM) outputs, given a HAP $\Psi = \langle \mathcal{M}, \mathcal{M}^H \rangle$ and the optimal solution π^* to $\mathcal{M}$, the *shortest* explanation (Chakraborti et al. 2017b) in the form of a model update to the human mental model $\mathcal{M}^H$ so that the same plan is now also optimal in the human's updated mental model $\widehat{\mathcal{M}}^H$ of the problem –

$$C(\pi^*, \widehat{\mathcal{M}}^H) = C^*_{\widehat{\mathcal{M}}^H}$$

The PEM can, in fact, trade off (Chakraborti, Sreedharan, and Kambhampati 2018) the relative cost of explicability (i.e. deviation from optimality in the planner's model) to the cost (i.e. length) of explanations during the plan generation process itself by computing a plan π and an explanation or model update $\mathcal{E}$ such that π is a solution to $\mathcal{M}$ and is the optimal solution to $\widehat{\mathcal{M}}$ modulated by a hyperparameter α –

$$\pi \leftarrow \arg\min_{\pi} \; |\mathcal{E}| \; + \; \alpha \times |\, C(\pi, \mathcal{M}) - C^*_{\mathcal{M}} \,|$$

With higher α, PEM computes plans that require more explanation, while with lower α, it generates more explicable plans. We refer to this variant as PEM(α).

Internally, PEM performs what is referred to as a *model space search* to come up with these explanations. This is done using *unit edit functions* λ that progressively try out one or more updates to the model $\mathcal{M}^H$ from the set of possible updates in $\mathcal{M} \Delta \mathcal{M}^H$ until the optimality conditions as described above are satisfied. This is known as the process of *model reconciliation* (Chakraborti et al. 2017b; Chakraborti, Sreedharan, and Kambhampati 2018).

3 ITS as Planning

We will now cast the design of a generic ITS in terms of the planning modules discussed in the previous section.

3.1 Class Configuration

A class configuration is defined as the tuple –

$$\mathcal{C} = \langle \{KC_i\}, \{T_i\}, \{A_i\}, \{S_i\} \rangle$$

- *Knowledge Components or Concepts:* $\{KC\}$ is a set of knowledge components or concepts KC_i. In ITS literature, the process of knowledge acquisition by a student has been decomposed into smaller components referred to as KCs (Koedinger, Corbett, and Perfetti 2010). KCs can be anything from a production rule (Mayer 1981), to a facet, misconception, fact or even a skill (Bloom, of College, and Examiners 1964). The aim of the social learning process is to make a student acquire different KCs based on their and their classmates already existing ones.
- *Tutorial:* The class also constitutes of a set $\{T_i\}$ of tutorials $T_i \subseteq \{KC_i\}$ that consist of a set of KCs on which they provide information on. These directly modify the student's knowledge state by providing information on specific topics or on how certain problems or (parts of) assignments may be solved. These form an integral part of a curriculum for the class.
- *Activities / Assignments:* The class also has a set $\{A_i\}$ of activities or assignments $A_i = \langle \mathcal{M}, \kappa \rangle$ where $\mathcal{M}$ is the model of the assignment and $\kappa \subseteq \{KC_i\}$ consists of a set of KCs that are required to solve it. These engage the student in actions that derive from knowledge introduced in the class (learning by doing). These form the core content of the class. Technically, these can also be used as sensing actions for the ITS in determining the knowledge state of the student. Thus, an assignment may be used both as a way of estimating the student model as well as a technique for imparting knowledge to the student.
- Finally, the class has a set $\{S_i\}$ of students S_i. The student knowledge state or *model* is defined as $S_i = \langle \{A_i^S\}, \kappa_1, \kappa_2 \rangle$ where A_i^S is the student's understanding (similar to the definition of a HAP) of the assignment model A_i and $\kappa_1, \kappa_2 \subseteq \{KC_i\}$ consists of a set of KCs that they have *learned* and *applied* respectively.

Given a class configuration $\mathcal{C}$, a curriculum is given by a sequence $c(\mathcal{C}) = \langle c_1, c_2, \ldots, c_n \rangle; c_i \in \{T_i\} \cup \{A_i\} \cup \{S_i\}$ of tutorials, assignments and partnerships with other students.

3.2 Tips and Hints

A solution to an assignment in a general sense can be seen as a sequence of steps, a.k.a. a *plan*. Thus, we posit that a large variety of assignments can in fact be modeled in terms of the planning problem. The model $A_i(\mathcal{M})$ of an assignment A_i (as mentioned before) is thus the model of a planning problem CPP. As explored in (Sengupta et al. 2017) in the context of decision support using automated planners, this opens up the slew of planning techniques (described in Section 2) that can be readily adopted to provide targeted (problem specific but domain independent) feedback to the students.

Solution Validation For a partial attempt (represented as a partial plan $\hat{\pi}$) on an assignment A_i, the Plan Validation Module (PVM) indicates conditions that were unsatisfied, which can be used to provide targeted feedback. For example, the PVM can be used by the instructor to auto-grade solutions proposed by a student, since this is a domain independent way of checking if the plan is a valid solution of the given assignment (represented as a CPP $A_i(\mathcal{M})$). This is also useful for the student as well who can receive immediate feedback on whether they are successful (and why, if not) without having to wait for the instructor. This is one of the features that most ITSs already possess. However, they are usually system level implementations that do not generalize across assignments.

Solution Completion For a partial attempt (represented as a partial plan $\hat{\pi}$) on an assignment A_i, the Plan Recognition Module (PRM) produces a completion that can be *sampled* from to provide hints that guide the student towards the full solution. The PRM thus allows the ITS to anticipate what actions the student needs to take given what they have already done in order to achieve their goal. Notice that the partial plan is generated by the student (from the model A_i^S) even though the completion is done using A_i. This can thus help the student in cases of cognitive overload, but not if they lack the knowledge to solve the problem, i.e. $A_i^S \neq A_i$. We will discuss ways to deal with the latter case in Sections 3.3.

The PRM module can be also used to provide *proactive support* by recognizing that the students is going astray and providing pop-ups to guide them towards the right solution. Proactive support and has been shown (Zhang et al. 2015; Sengupta et al. 2017; Chakraborti et al. 2017c) to be desirable of an artificial agent in collaborative settings. Interestingly, one could also imagine using the PRM to detect gaming of the tutoring system (Muldner et al. 2010) by defining it as a possible goal that a student might be trying to achieve, and based on the observations identify whether a student is working diligently or trying to game the system.

Problem Summarization Finally, the Landmark Generation Module (LGM) takes in the Classical Planning Problem (CPP) representation $A_i(\mathcal{M})$ for a specific assignment A_i and produces a set of steps (action landmarks) or situations (state landmarks) that the student *must* go through in order to solve the assignment. This can be very useful in providing a concise summary of "TODOs" required of the student to arrive at the solution, or by considering the domain variables that the student has already set to true, measure the progress of a student and thereby help the instructor in classroom orchestration (Dillenbourg et al. 2011).

We shall illustrate each of these use cases in Section 5.1.

3.3 On-demand Curriculum Generation

A typical feature of online learning, as we discussed in Section 1, is that students increasingly select a subset of class materials to follow and leave once they are done (e.g. MOOCs are known to have notoriously low completion rates (Amy Ahearn 2017)). As a result, students end up following individual and different curricula asynchronously. From the students' perspective an obvious problem with this is that they might not have the required knowledge to complete the materials they want. In the following, we thus address the problem of *on-demand curriculum generation*. In this paradigm, the student selects a particular assignment A_i to complete and the ITS performs argumentation with the assignment model $A_i(\mathcal{M})$ and the students model of the assignment $A_i^S(\mathcal{M})$ to identify deficiencies in the student model that need to be addressed using relevant tutorials.

In order to achieve this, the ITS spawns an instance of the Plan Explanation Module (PEM) with the HAP $\Psi = \langle A_i(\mathcal{M}), A_i^S(\mathcal{M})\rangle$ – here the instructor model is the ground truth and the student model needs to be reconciled. The model edit functions λ are the tutorials in the class. The output of the PEM is thus the *optimal set of tutorials* (this forms the recommended curriculum) that guarantees that the same solution (plan) is optimal in both the student model as well as the instructor model (even though they are not equal). This is especially useful since the instructor model is going to contain information pertaining to the entire class, while the student does not need to know all these details in order to solve a specific assignment. The PEM is thus able to leverage the student and instructor models of an assignment to provide the exact set of tutorials that the student requires. We will provide illustrations of this process in Section 5.2. Notice that, the ITS can either use its estimate of A_i^S or engage in active information gathering by asking the student questions to determine parts of the student model it is uncertain about (Sreedharan, Chakraborti, and Kambhampati 2018), in order to meet the specific needs of the student.

Teaching as an α trade-off Notice that the formulation of the assignments as planning problems allow us to spawn CCPs with the student models (indicating how the student can solve the problem) or the instructor model (indicating how the instructor will solve the same problem) or anywhere in between (as computed by PEM(α)). The student solution (equivalent to an explicable plan) is likely to be suboptimal, or in most cases, not feasible in the ground truth or instructor model. An instantiation of PEM(α) with the HAP $\Psi = \langle A_i(\mathcal{M}), A_i^S(\mathcal{M})\rangle$ thus allows us to modulate the level of expertise with which a student wants to solve an assignment. For low values of α, the ITS will recommend the smallest possible curriculum that will just enable the student to solve the assignment (albeit suboptimally) while for progressively higher values of α it will start recommending more and more advanced curriculum to the point it matches the output of PEM, i.e. the optimal complete curriculum. From the perspective of the instructor as well, the α hyperparameter can be gradually increase from a low value to generate study materials for individual students as the course progresses. Thus the teaching process itself can be viewed through the lens of the model reconciliation process as one of modulation of the value of α in the PEM(α). We shall demonstrate this in Section 5.2.

Remark To the best of our knowledge, algorithms for the on-demand curriculum generation process driven by a specific class activity, and the argumentation process over the curriculum with the desired expertise level of student, have not been explored before in the ITS literature. This technique can be useful from the perspective of both the instructor and the student – e.g. the former can stagger the course content to meet the student's expertise level, while the latter can chose to learn at different levels of expertise (thus possibly reducing the high dropout rates that plague the on-demand learning communities).

Composition of Student Models Finally, we note that we can extend the model edit functions in the PEM from just the tutorials in the class to the other student models as well. Thus the model updates during the model reconciliation process can be affected by either the KCs provided by tutorial or a composition of one or more student models. The output of PEM will now provide an optimal recommendation of tutorials and potential study partners based on the skill sets (i.e. models) of the individual students.

3.4 The Jigsaw Problem

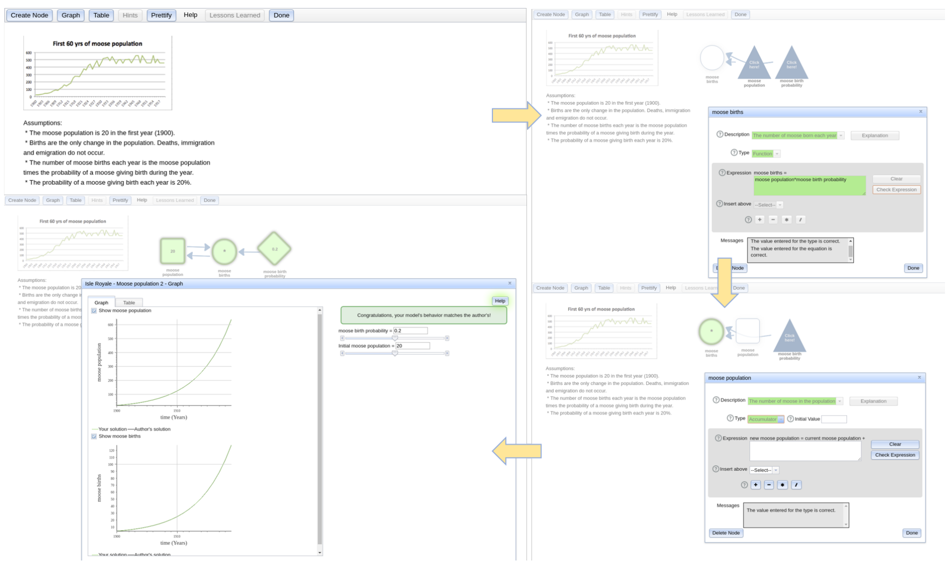
The Jigsaw Problem is the process of creating smaller groups in a class for cooperative learning (Aronson 1997). It has shown to have positive effect on students learning the course material together, and then engaging in discussions. This leads to a more active and deeper learning in class (Aronson 2011). Aronson, points out ten fixed steps to achieve this where the groups are created based on the ethnicity, race, gender and ability. However, it is intractable for a teacher to reason about all the student models and create study groups. Casting the class-level curriculum generation problem as a planning problem allows us to generate curricula for the entire class while enabling the instructor to specify desired properties of the curricula that needs to be maintained. These properties may be –

- Maximum size of study groups;
- Specific assignments of students;
- No repetition (or conversely, continuation) of study partners; and so on …
- In this paper, we specifically focus on the following property – *every student not only learns but applies all concepts in the class at least once*. This is especially important in the social learning paradigm, to ensure that students have mastered all concepts and not depended on other students to finish a shared curriculum.

In order to achieve this, we define a planning problem with the start state compiled from the class configuration $\mathcal{C}$ and a goal state that model a class configuration where $\forall S_i : S_i(\kappa_2) = \{KC_i\}$ – i.e. every student has applied all the concepts in class. The operators are generated from the set of tutorials and assignments – the tutorial operator has its associated KCs as effects of being learned; while assignment operators has KCs as preconditions (that need to be learned) and effects (of those KCs having been applied).

This formulation[3] thus not only ensures that all the students have mastered all the concepts in the class materials but also that the length of the curriculum is reduced (from $|\{S_i\}|$ times the length of the curriculum for individual students) due to the collaborations across students who can bring in complementary skill sets and transfer knowledge. We provide an illustration of this in Section 5.3.

[3]Note that this problem may be solved by horizon-limited planning, which is known to be NP-complete, the horizon being equal to $|\{S_i\}|$ times the length of the curriculum for individual students, which is the worst case curriculum length when no groups could be found. Thus, the jigsaw problem does not need the full expressiveness of CPP which is known to be PSPACE-complete.



Assumptions:
* The moose population is 20 in the first year (1900).
* Births are the only change in the population. Deaths, immigration and emigration do not occur.
* The number of moose births each year is the moose population times the probability of a moose giving birth during the year.
* The probability of a moose giving birth each year is 20%.

moose births
Description: The number of moose born each year
Type: Function
Expression: moose births = moose population*moose birth probability
Insert above: --Select--
Messages: The value entered for the type is correct. The value entered for the equation is correct.

moose population
Description: The number of moose in the population
Type: Accumulator
Initial Value:
Expression: new moose population = current moose population +
Insert above: --Select--
Messages: The value entered for the type is correct.

Isle Royale - Moose population 2 - Graph
Show moose population
Show moose births
Congratulations, your model's behavior matches the author's!
moose birth probability = 0.2
Initial moose population = 20

Figure 1: Illustration of the different stages of a "plan" being executed by a student in **Dragoon** – (1) the empty interface at the start of the problem (initial state); (2) the first node being completed; (2) the second node being created; and finally (3) the problem being completed with the feedback on the graph.

4 Introducing Dragoon

We will illustrate the above capabilities in **Dragoon** an ITS developed at Arizona State University to teach *dynamic system modeling* (VanLehn 2013) in the physical classroom setting – over the course of almost half a decade of deployment, the system has served 13 courses with approximate class sizes of 30, with more than a 1000 sessions per class. It is an ideal testbed for studying the nuances of tutoring systems currently deployed in classes in the space of mathematics, algebra and any other generic step-based tutoring systems. Figure 1 provides a snapshot of the interface.

In dynamic system modeling, a *system* is a part of the environment and *dynamic system* is the part of the environment that changes with time. Usually, first (or higher) order differential equations (differentiated with respect to time) represent dynamic systems mathematically. For simplicity of solving differential equations, time is discretized to calculate the values of different quantities. A *Model* refers to a representation of the system in a formal language.

Dragoon's formal language is based on Stella's stock and flow network (Doerr 1996). It consists of three different types of quantities – (1) **accumulator** (quantity that changes); (2) **function** (quantity that may or may not change); and (3) **parameter** (quantity that remains constant). These quantities are called *nodes*. To create a node a student needs to define its *properties* – i.e. description, type, value, units and equation. They are connected to each other by equations called *relations*. Students are taught template structure for interaction between nodes, which show particular rate of change in values called *schemas* – e.g. linear schema represents linear change in values while exponential schemas represent exponential changes. Students practice on **Dragoon** through tutorial and assignment workbooks. A detailed description of **Dragoon** is available at (Wetzel et al. 2017; VanLehn et al. 2016; 2017).

4.1 The Isle Royale Workbook

We use the Isle Royale Workbook (`https://goo.gl/ECrNnt`) to illustrate the proposed techniques. It teaches students population dynamics of moose and wolf population and learn interactions in a predator prey environment. There are six problems in the workbook (time step is a year) –

- **Isle-1** – Linear growth model of moose population, that is constant growth of two moose.
- **Isle-2** – Exponential growth model of moose population. The problem defines a constant growth rate which is multiplied by the population in the previous time-step to calculate the net growth.
- **Isle-3** – Exponential growth and death model of moose population. This problem adds the a constant death rate and the change in moose population is defined as the difference number of moose born and died.
- **Isle-4** – Exponential growth and death model of moose population with a fixed carrying capacity of the environment which effects the moose death rate.
- **Isle-5** – Exponential growth and death model of Wolf Population. This model is similar to Isle 3 problem.
- **Isle-6** – Exponential growth and death model of moose and wolf population with constant effect of wolf (predator) population on death rate of moose (prey) and constant effect of moose population on birth rate of wolf.

Epidemic schema is sometimes confused with exponential schema. Thus, we use one extra problem modeling flu epidemic in college which spreads through meetings between students. The number of students in the meeting and the chance that a student is affected is assumed to be constant.

The Zener Diode Problem Most problems in **Dragoon** are solved with a single or unique *set* of steps. The only thing that changes is the sequence in which nodes are created. However, there are a few problems which can be solved in multiple ways, where a student can change the equations in the nodes to solve the problem in lesser number of nodes. One such problem is to model a Zener diode using `Dragoon` – if a student has a more advanced understanding of circuit theory, then they can easily solve the problem in fewer steps (i.e. using fewer nodes). We will thus use this problem to demonstrate the usefulness of PEM(α).

5 ITS as Planning in Action

We will now illustrate how the techniques introduced in Section 3 manifests themselves on `Dragoon`. The first step is to construct the instructor model $\mathcal{M}^I$ – examples can be accessed at – `https://goo.gl/cyVthK`.

We used nested object types to represent different objects in Dragoon, i.e. node, schema (KCs) and properties. Accumulator, parameter and function were of type node. Linear, exponential, extended_exponential, carrying_capacity and epidemic were types of schema. Description, value, type, equation and units are type of properties. These object types were used to define the state variables which characterize the properties that were part of a node, nodes that were part of schema, and schemas that were part of the problem. The operators in the domain represent the actions that are available student in the `Dragoon` environment. For example, a student fills each property to complete a node and it can be done in a fixed order. So the operator definitions were also related to initializing a node, filling every property of the node, completing a node and completing a schema. Students need an understanding of the schema to fill the type and equation of the node. Thus actions for those steps have a precondition of has_schema to create the node. Finally, the initial state consists of all the nodes and schemas that are part of the assignment as well as the knowledge state of the student, that is whether they understand the schemas required to solve the problem. The goal state required that the student complete all the schemas that are present in a given problem.

5.1 Tips and Hints (c.f. Section 3.2)

Plan Validation Figure 2, shows the 20-step solution for **Isle-2**, and Figure 1 shows some of these actions in the `Dragoon` environment. Figure 2 presents the incomplete attempt of the student being flagged as unsuccessful by the PVM, and shows the error generated after executing the incomplete plan in the `Dragoon` interface.

Plan Recognition Figure 3 shows the correct identification by the PRM among two possible solutions of the **Isle-3** assignment using the "exponential_growth" schema or the "exponential_decay" schema from partial observations of the actions of the student in `Dragoon`.

Landmarks Figure 4 shows the 35 state landmarks produced by the LGM for the **Isle-3** assignment.

5.2 On-demand Curriculum Generation (c.f. Section 3.3)

We use the same domain that we used in tips and hints. We are testing the case where a student wants to solve the **Isle-4** problem. Figure 5 shows the output of PEM when a student expresses a desire to complete the **Isle-4** assignment and requests a curriculum for it. The explanation presents the model differences in the initial state that prevents the student from completing the assignment at this time and suggests tutorials to introduce these concepts. The explanation is of size 3, and references the missing knowledge concepts that are needed for solving the problem in the 40 steps.

Figure 6 shows how PEM(α) can be used to modulate the expertise levels of the recommended curriculum. The complete curriculum is of size 3 after which the problem can be solved in 17 steps. But, with lower value of α, the problem can be solved with a longer 20 step plan. As explained earlier, even though the student needs two knowledge concepts to solve the problem (zener_voltage_regulator and kvl schema), but to solve the optimal plan a student needs to be an expert and improve the equations in one of the nodes and create a better model for Zener Diode problem.

5.3 Jigsaw Problem (c.f. Section 3.4)

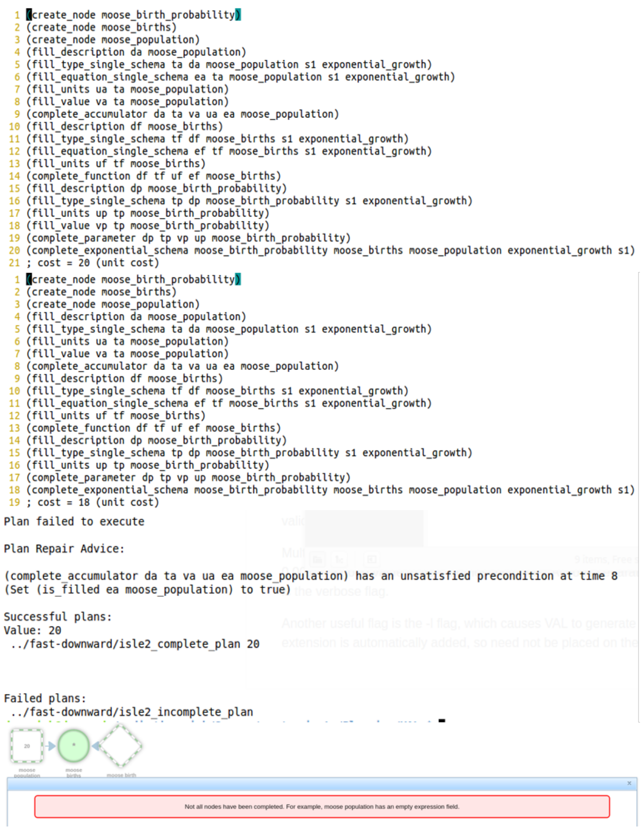
Here, we took an instance of a `Dragoon` class with 7 concepts and 9 assignments. A single student curriculum comes out as 12 steps long, with 7 tutorials and 5 assignments. However, with the introduction of groups of two students, this reduces to a combined curriculum of 23 steps where every student applies every concept at least once. For every new student, plan size increases by 11 steps, showing that one of the assignment can be done in the group. This is shown in Figure 7, which plots the curriculum length with increasing class sizes. In this particular class configuration, only one of the assignments could be done in a group.

Now we study the effect of varying class configurations by the making assignments that randomly teach up to 4 concepts. The number of concepts were fixed to 10 and there were 20 assignments that would teach these concepts. Figure 8 shows the curriculum length for 50 different randomly generated four student class configurations. We observe a decrease of 3 to 7 steps in every class.

Conclusion and Work in Progress

In this paper, we demonstrated how an ITS framework can be built using the state-of-the-art in human-aware planning techniques for the design of course independent support features. The last section illustrated these properties in a real tutoring system `Dragoon`. Currently, we are working on integrating these features into `Dragoon` in order to perform ablation studies of the system with one or more of the support components deployed in a real class. We hope to report on those results in future iterations of the paper.

Acknowledgements. This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014-16-1-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, N00014-18-1-2442 and the NASA grant NNX17AD06G. The first author is also supported by the IBM Ph.D. Fellowship from 2016-18. The authors also thank Professors Kurt Van Lehn and Erin Walker (Arizona State University) for their valuable inputs.



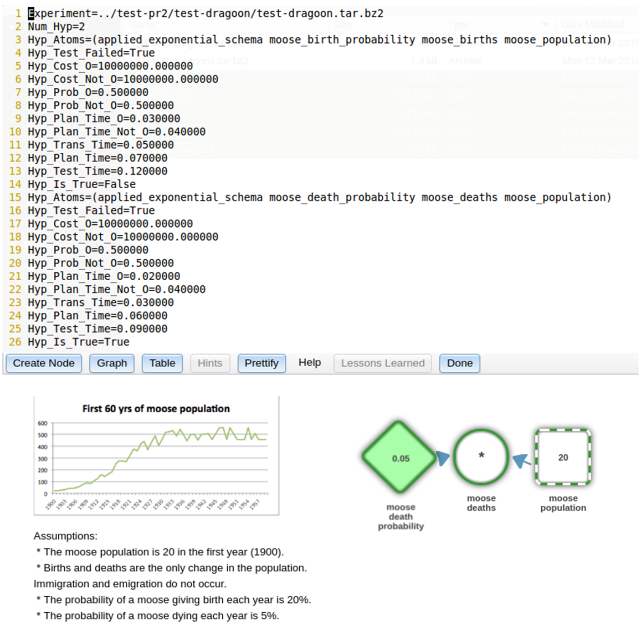
```
(create_node moose_birth_probability)
(create_node moose_births)
(create_node moose_population)
(fill_description da moose_population)
(fill_type_single_schema ta da moose_population s1 exponential_growth)
(fill_equation_single_schema ea ta moose_population s1 exponential_growth)
(fill_units ua ta moose_population)
(fill_value va ta moose_population)
(complete_accumulator da ta va ua ea moose_population)
(fill_description df moose_births)
(fill_type_single_schema tf df moose_births s1 exponential_growth)
(fill_equation_single_schema ef tf moose_births s1 exponential_growth)
(fill_units uf tf moose_births)
(complete_function df tf uf ef moose_births)
(fill_description dp moose_birth_probability)
(fill_type_single_schema tp dp moose_birth_probability s1 exponential_growth)
(fill_units up tp moose_birth_probability)
(fill_value vp tp moose_birth_probability)
(complete_parameter dp tp vp up moose_birth_probability)
(complete_exponential_schema moose_birth_probability moose_births moose_population exponential_growth s1)
; cost = 20 (unit cost)
```

```
(create_node moose_birth_probability)
(create_node moose_births)
(create_node moose_population)
(fill_description da moose_population)
(fill_type_single_schema ta da moose_population s1 exponential_growth)
(fill_units ua ta moose_population)
(fill_value va ta moose_population)
(complete_accumulator da ta va ua ea moose_population)
(fill_description df moose_births)
(fill_type_single_schema tf df moose_births s1 exponential_growth)
(fill_equation_single_schema ef tf moose_births s1 exponential_growth)
(fill_units uf tf moose_births)
(complete_function df tf uf ef moose_births)
(fill_description dp moose_birth_probability)
(fill_type_single_schema tp dp moose_birth_probability s1 exponential_growth)
(fill_units up tp moose_birth_probability)
(complete_parameter dp tp vp up moose_birth_probability)
(complete_exponential_schema moose_birth_probability moose_births moose_population exponential_growth s1)
; cost = 18 (unit cost)
```

```
Plan failed to execute

Plan Repair Advice:

(complete_accumulator da ta va ua ea moose_population) has an unsatisfied precondition at time 8
(Set (is_filled ea moose_population) to true)

Successful plans:
Value: 20
 ../fast-downward/isle2_complete_plan 20



Failed plans:
 ../fast-downward/isle2_incomplete_plan
```

Not all nodes have been completed. For example, moose population has an empty expression field.

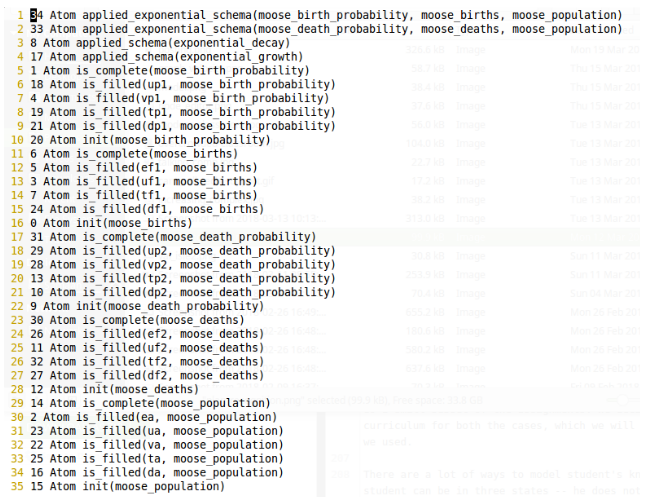
Figure 2: Response of PVM to the correct and incorrect or incomplete attempts in the **Isle-3** problem.



```
Experiment=../test-pr2/test-dragoon/test-dragoon.tar.bz2
Num_Hyp=2
Hyp_Atoms=(applied_exponential_schema moose_birth_probability moose_births moose_population)
Hyp_Test_Failed=True
Hyp_Cost_O=10000000.000000
Hyp_Cost_Not_O=10000000.000000
Hyp_Prob_O=0.500000
Hyp_Prob_Not_O=0.500000
Hyp_Plan_Time_O=0.030000
Hyp_Plan_Time_Not_O=0.040000
Hyp_Trans_Time=0.050000
Hyp_Plan_Time=0.070000
Hyp_Test_Time=0.120000
Hyp_Is_True=False
Hyp_Atoms=(applied_exponential_schema moose_death_probability moose_deaths moose_population)
Hyp_Test_Failed=True
Hyp_Cost_O=10000000.000000
Hyp_Cost_Not_O=10000000.000000
Hyp_Prob_O=0.500000
Hyp_Prob_Not_O=0.500000
Hyp_Plan_Time_O=0.020000
Hyp_Plan_Time_Not_O=0.040000
Hyp_Trans_Time=0.030000
Hyp_Plan_Time=0.060000
Hyp_Test_Time=0.090000
Hyp_Is_True=True
```

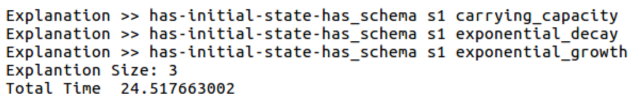
Assumptions:
 * The moose population is 20 in the first year (1900).
 * Births and deaths are the only change in the population.
Immigration and emigration do not occur.
 * The probability of a moose giving birth each year is 20%.
 * The probability of a moose dying each year is 5%.

Figure 3: The output of the PRM in the **Isle-3** problem which can be solved in two separate ways. Here the student seemed to have decided to work on the exponential_decay schema.



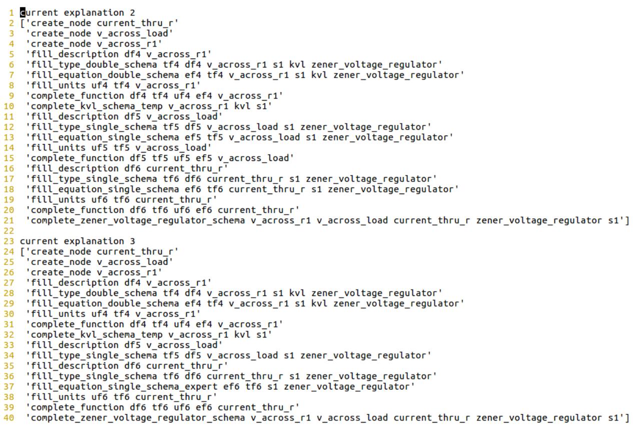
```
34 Atom applied_exponential_schema(moose_birth_probability, moose_births, moose_population)
33 Atom applied_exponential_schema(moose_death_probability, moose_deaths, moose_population)
8 Atom applied_schema(exponential_decay)
17 Atom applied_schema(exponential_growth)
1 Atom is_complete(moose_birth_probability)
18 Atom is_filled(up1, moose_birth_probability)
4 Atom is_filled(vp1, moose_birth_probability)
19 Atom is_filled(tp1, moose_birth_probability)
21 Atom is_filled(dp1, moose_birth_probability)
20 Atom init(moose_birth_probability)
6 Atom is_complete(moose_births)
5 Atom is_filled(ef1, moose_births)
3 Atom is_filled(uf1, moose_births)
7 Atom is_filled(tf1, moose_births)
24 Atom is_filled(df1, moose_births)
0 Atom init(moose_births)
31 Atom is_complete(moose_death_probability)
29 Atom is_filled(up2, moose_death_probability)
28 Atom is_filled(vp2, moose_death_probability)
13 Atom is_filled(tp2, moose_death_probability)
10 Atom is_filled(dp2, moose_death_probability)
9 Atom init(moose_death_probability)
30 Atom is_complete(moose_deaths)
26 Atom is_filled(ef2, moose_deaths)
11 Atom is_filled(uf2, moose_deaths)
32 Atom is_filled(tf2, moose_deaths)
27 Atom is_filled(df2, moose_deaths)
12 Atom init(moose_deaths)
14 Atom is_complete(moose_population)
2 Atom is_filled(ea, moose_population)
23 Atom is_filled(ua, moose_population)
22 Atom is_filled(va, moose_population)
25 Atom is_filled(ta, moose_population)
16 Atom is_filled(da, moose_population)
15 Atom init(moose_population)
```

Figure 4: The 35 state landmarks generated by the LGM for the **Isle-3** problem.



```
Explanation >> has-initial-state-has_schema s1 carrying_capacity
Explanation >> has-initial-state-has_schema s1 exponential_decay
Explanation >> has-initial-state-has_schema s1 exponential_growth
Explantion Size: 3
Total Time  24.517663002
```

Figure 5: On-demand curriculum generated by the PEM. This is the smallest change to the student model required to solve the **Isle-4** problem.



```
current explanation 2
['create_node current_thru_r'
 'create_node v_across_load'
 'create_node v_across_r1'
 'fill_description df4 v_across_r1'
 'fill_type_double_schema tf4 df4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_equation_double_schema ef4 tf4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_units uf4 tf4 v_across_r1'
 'complete_function df4 tf4 uf4 ef4 v_across_r1'
 'complete_kvl_schema_temp v_across_r1 kvl s1'
 'fill_description df5 v_across_load'
 'fill_type_single_schema tf5 df5 v_across_load s1 zener_voltage_regulator'
 'fill_equation_single_schema ef5 tf5 v_across_load s1 zener_voltage_regulator'
 'fill_units uf5 tf5 v_across_load'
 'complete_function df5 tf5 uf5 ef5 v_across_load'
 'fill_description df6 current_thru_r'
 'fill_type_single_schema tf6 df6 current_thru_r s1 zener_voltage_regulator'
 'fill_equation_single_schema ef6 tf6 current_thru_r s1 zener_voltage_regulator'
 'fill_units uf6 tf6 current_thru_r'
 'complete_function df6 tf6 uf6 ef6 current_thru_r'
 'complete_zener_voltage_regulator_schema v_across_r1 v_across_load current_thru_r zener_voltage_regulator s1']

current explanation 3
['create_node current_thru_r'
 'create_node v_across_load'
 'create_node v_across_r1'
 'fill_description df4 v_across_r1'
 'fill_type_double_schema tf4 df4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_equation_double_schema ef4 tf4 v_across_r1 s1 kvl zener_voltage_regulator'
 'fill_units uf4 tf4 v_across_r1'
 'complete_function df4 tf4 uf4 ef4 v_across_r1'
 'complete_kvl_schema_temp v_across_r1 kvl s1'
 'fill_description df5 v_across_load'
 'fill_type_single_schema tf5 df5 v_across_load s1 zener_voltage_regulator'
 'fill_description df6 current_thru_r'
 'fill_type_single_schema tf6 df6 current_thru_r s1 zener_voltage_regulator'
 'fill_equation_single_schema_expert ef6 tf6 s1 zener_voltage_regulator'
 'fill_units uf6 tf6 current_thru_r'
 'complete_function df6 tf6 uf6 ef6 current_thru_r'
 'complete_zener_voltage_regulator_schema v_across_r1 v_across_load current_thru_r zener_voltage_regulator s1']
```

Figure 6: Different plans and associated model updates generated by the PEM(α) based on the α-hyperparameter. For a high value of α the curriculum is of size 3 after which the problem can be solved in 17 steps. With a lower value of α, the problem can be solved with a longer 20 step plan.

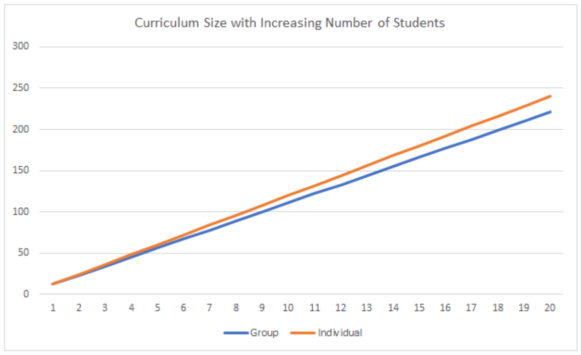


Figure 7: Group versus individual curriculum lengths with increasing class size.

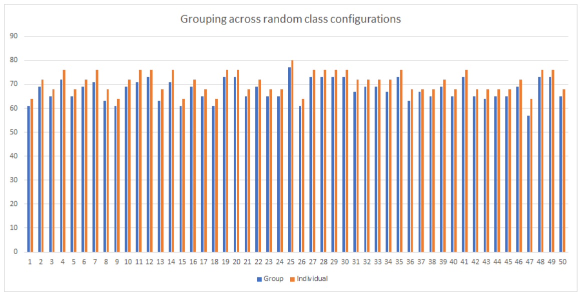


Figure 8: Group versus individual curriculum lengths in different class configurations.

References

Amy Ahearn. 2017. The Flip Side of Abysmal MOOC Completion Rates? Discovering the Most Tenacious Learners. `https://goo.gl/DR7nxa`. EdSurge.

Aronson, E. 1997. *The jigsaw classroom: Building cooperation in the classroom*. Scott Foresman & Company.

Aronson, E. 2011. *Cooperation in the classroom: The jigsaw method*. Printer & Martin Limited.

Barnes, T., and Stamper, J. 2010. Automatic hint generation for logic proof tutoring using historical data. *Journal of Educational Technology & Society* 13(1):3.

Bloom, B. S.; of College, C.; and Examiners, U. 1964. *Taxonomy of educational objectives*, volume 2. Longmans, Green New York.

Bouchet, F.; Labarthe, H.; Yacef, K.; and Bachelet, R. 2017. Comparing peer recommendation strategies in a mooc. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, 129–134. ACM.

Burke, A. 2011. Group work: How to use groups effectively. *Journal of Effective Teaching* 11(2):87–95.

Chakraborti, T.; Kambhampati, S.; Scheutz, M.; and Zhang, Y. 2017a. AI Challenges in Human-Robot Cognitive Teaming. *arXiv preprint arXiv:1707.04775*.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017b. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*.

Chakraborti, T.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2017c. Mr.Jones – Towards a Proactive Smart Room Orchestrator. In *AAAI Fall Symposium on Human-Agent Groups*.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2018. Balancing Explicability and Explanation in Human-Aware Planning. In *AAMAS*.

Chi, M.; VanLehn, K.; and Litman, D. 2010. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *ITS*, 224–234.

Connelly, J., and Katz, S. 2009. Toward more robust learning of physics via reflective dialogue extensions. In *EdMedia: World Conference on Educational Media and Technology*, 1946–1951. Association for the Advancement of Computing in Education (AACE).

Corbett, A. T., and Anderson, J. R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4(4):253–278.

Dillenbourg, P.; Zufferey, G.; Alavi, H.; Jermann, P.; Do-Lenh, S.; Bonnard, Q.; Cuendet, S.; and Kaplan, F. 2011. Classroom orchestration: The third circle of usability. *CSCL2011 Proceedings* 1:510–517.

Doerr, H. M. 1996. Stella ten years later: A review of the literature. *International Journal of Computers for Mathematical Learning* 1(2):201–224.

Gertner, A. S., and VanLehn, K. 2000. Andes: A coached problem solving environment for physics. In *International conference on intelligent tutoring systems*, 133–142.

Graesser, A. C.; Chipman, P.; Haynes, B. C.; and Olney, A. 2005. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*.

Hambleton, R. K.; Swaminathan, H.; and Rogers, H. J. 1991. *Fundamentals of item response theory*, volume 2. Sage.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR*.

Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *Tools with Artificial Intelligence (ICTAI). 16th IEEE International Conference on*, 294–301. IEEE.

Kambhampati, S., and Talamadupula, K. 2015. Human-in-the-loop planning and decision support. *AAAI Tutorial*.

Katz, S.; Allbritton, D.; and Connelly, J. 2003. Going beyond the problem given: How human tutors use post-solution discussions to support transfer. *International Journal of Artificial Intelligence in Education* 13(1):79–116.

Katz, S.; O'Donnell, G.; and Kay, H. 2000. An approach to analyzing the role and structure of reflective dialogue. *International Journal of Artificial Intelligence in Education*.

Koedinger, K. R.; Corbett, A. T.; and Perfetti, C. 2010. The knowledge-learning-instruction (kli) framework: Toward bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*.

Kulkarni, A.; Chakraborti, T.; Zha, Y.; Vadlamudi, S. G.; Zhang, Y.; and Kambhampati, S. 2016. Explicable Robot

Planning as Minimizing Distance from Expected Behavior. *CoRR* abs/1611.05497.

Labarthe, H.; Bouchet, F.; Bachelet, R.; and Yacef, K. 2016. Does a peer recommender foster students' engagement in moocs? In *9th International Conference on Educational Data Mining*, 418–423.

Magnisalis, I.; Demetriadis, S.; and Karakostas, A. 2011. Adaptive and intelligent systems for collaborative learning support: A review of the field. *IEEE transactions on Learning Technologies* 4(1):5–20.

Mandel, T.; Liu, Y.-E.; Levine, S.; Brunskill, E.; and Popovic, Z. 2014. Offline policy evaluation across representations with applications to educational games. In *AAMAS*.

Mandel, T. 2017. Refraction: Teaching Fractions through Gameplay. `https://goo.gl/BrPpmV`.

Mayer, R. E. 1981. Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *Instructional Science* 10(2):135–175.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language.

McLoughlin, C., and Lee, M. J. 2007. Social software and participatory learning: Pedagogical choices with technology affordances in the web 2.0 era. In *Proceedings of ICT: Providing choices for learners and learning*, 664–675.

Mesch, D. J. 1991. The jigsaw technique: A way to establish individual accountability in group work. *Journal of Management Education* 15(3):355–358.

Mitrovic, A. 2003. An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*.

Muldner, K.; Burleson, W.; Van de Sande, B.; and VanLehn, K. 2010. An analysis of gaming behaviors in an intelligent tutoring system. In *International Conference on Intelligent Tutoring Systems*, 184–193. Springer.

Murray, R., and VanLehn, K. 2006. A comparison of decision-theoretic, fixed-policy and random tutorial action selection. In *Intelligent Tutoring Systems*, 114–123.

Murray, R. C.; VanLehn, K.; and Mostow, J. 2004. Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education* 14(3, 4):235–278.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *IJCAI*.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Rivers, K., and Koedinger, K. R. 2013. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, volume 50.

Rivers, K., and Koedinger, K. R. 2017. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education* 27(1):37–64.

Russell, S., and Norvig, P. 2003. *Artificial intelligence: a modern approach*. Prentice Hall.

Scardamalia, M., and Bereiter, C. 2006. Knowledge building: Theory, pedagogy, and technology. *The Cambridge Handbook of the Learning Sciences* 97–118.

Schulze, K. G.; Shelby, R. N.; Treacy, D. J.; Wintersgill, M.; VanLehn, K.; and Gertner, A. 2000. Andes: An active learning, intelligent tutoring system for newtonian physics. *THEMES in Education* 1(2):115–136.

Seely Brown, J., and Adler, R. 2008. Open education, the long tail, and learning 2.0. *Educause review* 43(1):16–20.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2017. RADAR – A Proactive Decision Support System for Human-in-the-Loop Planning. In *AAAI Fall Symposium on Human-Agent Groups*.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*.

Stamper, J.; Eagle, M.; Barnes, T.; and Croy, M. 2013. Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education* 22(1-2):3–17.

VanLehn, K.; Chung, G.; Grover, S.; Madni, A.; and Wetzel, J. 2016. Learning science by constructing models: Can dragoon increase learning without increasing the time required? *International Journal of Artificial Intelligence in Education*.

VanLehn, K.; Wetzel, J.; Grover, S.; and van de Sande, B. 2017. Learning how to construct models of dynamic systems: An initial evaluation of the dragoon intelligent tutoring system. *IEEE Transactions on Learning Technologies*.

Vanlehn, K. 2006. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*.

VanLehn, K. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist* 46(4):197–221.

VanLehn, K. 2013. Model construction as a learning activity: A design space and review. *Interactive Learning Environments* 21(4):371–413.

Webb, N. M. 2013. Information processing approaches to collaborative learning. *Routledge Handbooks Online*.

Wetzel, J.; VanLehn, K.; Butler, D.; Chaudhari, P.; Desai, A.; Feng, J.; Grover, S.; Joiner, R.; Kong-Sivert, M.; Patade, V.; et al. 2017. The design and development of the dragoon intelligent tutoring system for model construction: Lessons learned. *Interactive Learning Environments* 25(3):361–381.

Zhang, Y.; Narayanan, V.; Chakraborty, T.; and Kambhampati, S. 2015. A human factors analysis of proactive assistance in human-robot teaming. In *IROS*.

Zhang, Y.; Sreedharan; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2016. Plan Explicability for Robot Task Planning. In *RSS Workshop on Planning for Human-Robot Interaction*.

Zhang, Y.; Sreedharan; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan Explicability and Predictability for Robot Task Planning. In *ICRA*.

Zhu, L., and Givan, R. 2003. Landmark extraction via planning graph propagation. *ICAPS Doctoral Consortium*.