Data-driven Training Development: Deriving Performance Constraints from Operational Examples

Randy Jensen, Sowmya Ramachandran
Stottler Henke Associates, Inc.
San Mateo, CA
jensen@shai.com, sowmya@shai.com

ABSTRACT

Many modern operational performance environments produce significant data artifacts that collectively constitute rich libraries of decision-making examples. For domains where expert decisions are guided by constraints, there is the potential to automatically derive the constraints themselves from expert performance data. This paper discusses a data-driven machine learning approach to modeling constraints, implemented in an authoring tool coupled with a simulation-based training environment for satellite planners. In this domain, the planner's task is to create a 7-day schedule of requested satellite contacts, while meeting a range of specialized planning constraints which vary for different satellites with different missions. The training goal is to assess planners' decisions in simulation-based scenarios and provide feedback, which requires automated performance assessment measures with knowledge of planning constraints. For this application, the authoring tool provides a utility to directly process operational source data, in this case consisting of archived records of satellite requests from previous periods. This produces derived constraints, which authors then review, edit, and annotate as needed before linking the constraints to runtime assessment mechanisms for exercises. Beyond the initial focus on generating automated assessment with this data-driven approach, the development process uncovered other useful applications for the ability to derive constraints from operational data. For example, one phase of the satellite planning process involves deconflicting one's own satellite support requests from those involving other satellites that may be seeking to simultaneously use the same resources, such as a specific ground antenna. In order to support individual training for this task, an automated agent was created to produce realistic simulated conflicts with the planner's requests, based on constraints mined from operational data. This research also helped to uncover drawbacks in the data-driven approach for some domains, so this paper discusses applicability and limitations in more general cases beyond the initial satellite planning application.

ABOUT THE AUTHORS

Randy Jensen is a group manager at Stottler Henke Associates, Inc., working in training systems since 1993. His research areas include adaptive training, distributed learning, game-based training, behavior modeling, and natural language processing. He has led projects to develop Intelligent Tutoring Systems and automated after action review tools for the Army, Air Force, Navy, and Marines. Recent work includes a project for the U.S. Air Force to use machine learning techniques to generate performance assessment mechanisms from operational data. He also recently led the development of model-based performance assessment in an intelligent tutor for training troubleshooting skills for the U.S. Navy. Mr. Jensen holds a B.S. with honors in symbolic systems from Stanford University.

Dr. Sowmya Ramachandran is a research scientist at Stottler Henke Associates, where her research focuses on the application of AI and Machine Learning to improve education and training. She leads research and development of intelligent tutoring systems (ITSs) and ITS authoring tools for a diverse range of military and civilian domains. Dr. Ramachandran headed the development of ReadInsight, an intelligent tutor for teaching reading comprehension skills to adult English speakers. She also led the development of a tutor for training Tactical Action Officers in the Navy. This system uses natural language processing technologies to assess and train TAOs and is currently in operational use at the Surface Warfare Officers School. She has recently led the development of an ITS for training U.S. Navy Information Systems Technicians in troubleshooting and maintenance skills. Dr. Ramachandran holds a Ph.D. from The University of Texas at Austin. For her dissertation, she developed a novel machine learning technique for constructing Bayesian Network models from data.

Data-driven Training Development: Deriving Performance Constraints from Operational Examples

Randy Jensen, Sowmya Ramachandran
Stottler Henke Associates, Inc.
San Mateo, CA
jensen@shai.com, sowmya@shai.com

INTRODUCTION

In recent years, machine learning methods have been increasingly employed with large data sets to produce solutions for problems that would otherwise require significant work codifying the knowledge of human experts. Even as data-driven approaches become more common, there are still many untapped resources in the form of existing data that could be mined for practical applications. In simulation-based training, data products often conform to the same formats and protocols as the performance environments that are their operational counterparts. Thus operational data sets can collectively constitute rich libraries of decision-making examples with a range of potential uses in training. For domains where expert decisions are guided by constraints, there is the potential to automatically derive the constraints themselves from expert performance data, as an alternative to manually encoding constraint knowledge. This paper discusses a data-driven machine learning approach to modeling constraints, implemented in an authoring tool that processes historical operational data, for execution in a simulation-based training environment.

For this application, the primary objective for deriving constraints is to generate automated performance assessment functionality that can be executed by an Intelligent Tutoring System (ITS) integrated with a training simulation. ITS technologies augment or emulate the role of human instructors to support training through experiential practice, with individualized guidance and feedback. They can be powerful training tools, and have demonstrated effectiveness in producing meaningful learner effects (VanLehn, 2011; Ma, Adescope, Nesbit, & Liu, 2014; Kulik & Fletcher, 2016). One of the most complex components to develop in an ITS is the learner model, which supports the process of closely monitoring a student in an exercise context to assess their performance and provide context-sensitive, personalized coaching (Woolf, 2008). We describe an authoring approach that focuses on the learner modeling process, with a mechanism to build assessment models from source data in the operational domain, using domain-standard file input/output utilities and categorical constraint definitions to parse data.

The context for the authoring tool is a test case application with the task domain of satellite planning. In this domain, the planner's task is to create a 7-day schedule of requested satellite supports, which are time-bounded communication intervals involving a pairing of an individual satellite with an available ground antenna. In building their schedules, planners must meet a range of specialized planning constraints which vary for different satellites with different missions. The training goal in this application is to assess planners' decisions in simulation-based scenarios and provide feedback, using automated performance assessment measures with knowledge of both general and satellite-specific planning constraints. The source data used to derive constraints consists of archived records of satellite requests from previous periods. Once constraints are generated, it's important to have an annotation tool where authors can review and edit as needed before linking the constraints to runtime assessment mechanisms for exercises, so that authors have final control over how assessments work in training.

This paper describes the data-driven authoring method and the specific application implemented for training satellite planners, as well as findings about strengths and drawbacks in this approach. One noteworthy finding that emerged from the development process is that there are numerous useful applications for performance constraints derived from operational data, that may not necessarily be anticipated with the initial implementation of a constraint authoring capability. For example, although the test case application for the authoring tool initially focused on generating constraints for automated assessment capabilities associated with learner modeling, we also observed that the same method could be productively used to derive constraints for automated roleplayer behaviors to make exercises realistic and adaptive. Data-driven methods can also face inherent challenges as well, such as noise in the data, insufficient samples, and any hidden factors that are not captured in the data despite playing a role in the decisions that are reflected in the data. This paper reviews some of these challenges and how they ultimately relate to the determination of whether data-driven authoring is effective for a given training application.

BACKGROUND

Constraint-based modeling is one of an array of learner modeling techniques used in intelligent tutors, each with different advantages varying with the nature of the knowledge, skills, and performance environment associated with the domain. One factor that makes the constraint-based approach to representation of assessment knowledge effective for some domains is a degree of flexibility that it provides. Whereas some rule-based assessment mechanisms require very specific and detailed domain knowledge, constraints can be as specific or as general as the training task demands. Thus, highly specific constraints can be developed for a well-defined topic like Algebra or Physics. On the other hand, somewhat general constraints can be used to provide fairly granular assessments for domains that involve more flexibility in the solutions, such as tactical decision making. Constraint-based models can also be flexibly used for different types of training tasks such as decision-making or procedural tasks. There has been some prior work using data-driven methods for learner model authoring; for example, (McLaren, Koedinger, Schneider, Harrer, & Bollen, 2004) used a technique called bootstrapping novice data to derive production rules for correct and buggy actions from actual student logs. With the application of data-driven methods to constraint-based modeling, this effort attempts to build on such prior work.

In general, when constraints are used to represent performance knowledge, they are constructed with two conditions: a relevance condition, and a success condition (Mitrovic, Martin, & Suraweera, 2007). The relevance condition specifies when a constraint is applicable. The relevance condition of constraints can be used to limit the scope of assessment in this manner. Success conditions are those that must be met in order for the constraint to be satisfied. These can be statements of actions that must be performed, or simulation states that must be created in order to satisfy the constraint. When a constraint is satisfied, the student's performance with respect to that constraint is assessed a success. Violations of constraints lead to negative assessments.

In our test case training application for the satellite planning domain, the student's actions in a training exercise occur as discrete events, for example when initially submitting a planning solution. Therefore for the assessment constraints, relevance conditions are not concerned with timing, but rather with context within a planning solution. At the highest level, the satellite planning tasks involve the goal of finding an optimal schedule of resource allocations while considering tradeoffs between various scheduling parameters. Resources include the satellites themselves, and ground antennas that they will communicate with. Allocations depend on satellite mission requirements, and compatibility in terms of visibility times between satellites and antennas, as well as certain technical factors involved in communications. Availabilities and visibilities continually change from one period to the next, based on orbital perturbations, maintenance intervals, and changing mission needs. So the planning task, either in the operational setting or in a training exercise, involves finding a solution that navigates between different tradeoffs, all within the confines of the unique conditions that apply for the planning period. A simple example of a scheduling parameter is the minimum duration for a support, as required to satisfy the mission requirements or purpose of the support. There may be an available time window with a visibility between the satellite and antenna, but if the duration of the visibility is too short, then it is not a suitable choice for the planner's schedule. In this application, constraints are the numerical values associated with parameters that establish the band of tolerance for effective supports in accord with mission requirements. For each satellite, mission requirements are documented, and also generally known by expert planners. This provided a testable research question for the development of a data-driven authoring tool:

> *Given direct knowledge of satellite-specific planning constraints (as described in documentation), can planning constraints be derived from historical data to effectively mimic known constraints, for performance assessment in training?*

This question was tested by implementing the authoring tool, gathering sample data, deriving constraints, employing the derived constraints in a simulation-based tutor, and evaluating both the constraints and the tutor with end users at a satellite operations center. For subjective overall feedback on the authored assessments and the tutor, only two end users were available to evaluate the system, partly due to the specialized expertise involved in their roles. More objective findings were collected for the main question about how well the derived constraints match documented mission requirements, by reviewing nominal vs. derived values for the constraints. While acknowledging that this test case application is a single instance, we can report that both the subjective and objective findings were favorable with regard to the feasibility of using derived performance constraints for assessment.

Task Background – Satellite Planning

Due to the subject matter, this paper does not go into detail about satellite planning tasks as performed in either the operational or training setting. By the same token, details are also omitted for the mission-related constraints associated with either specific satellites or satellite planning in general (with the exception of one example constraint mentioned earlier, involving support duration). Thus, domain examples will be limited. At a high level, there are two satellite planning tasks in the test case application:

- Initial planning. This involves constructing a plan for a collection of satellite support requests in a planning period, for a set of one or more specific satellites. The plan is then submitted to a central scheduling organization that aggregates *all* requests for *all* satellites. The process of developing the initial plan involves decisions related to mission constraints for different satellites, based on information about orbital visibilities and resource availability in the planning period. At this step, notional availability does not include information about the concurrent requests from other planners responsible for other satellites. So the initial plans for different satellites regularly contain overlaps in requested resources and times (e.g., using the same antenna at the same time). From an individual satellite planner's task perspective, the objective in developing the initial plan primarily relates to satisfying mission constraints for their own satellites, without attempting any coordination with other planners.
- Deconfliction. After the initial plan is submitted, there is a deconfliction process where the central scheduling authority collects all requests, and then disseminates a proposed global deconflicted schedule back to the individual satellite planners, who make final decisions. Since the initial requests for different satellites may have conflicts, the deconflicted schedule often includes requested changes which the satellite planners must review in light of their own planning constraints. For the satellite planners, the deconfliction task is similar to initial planning in terms of the relevant constraints and the overall task objective. However, the constraints are slightly different due to the practical tradeoffs that come into play when conflicts must be resolved. There are also cases of additional constraints that are only considered at the final step of deconfliction.

In the simulation-based trainer, exercise flow mimics the operational planning flow from the initial stage to deconfliction, although the timeline between these tasks is artificially accelerated compared to real-world operations.

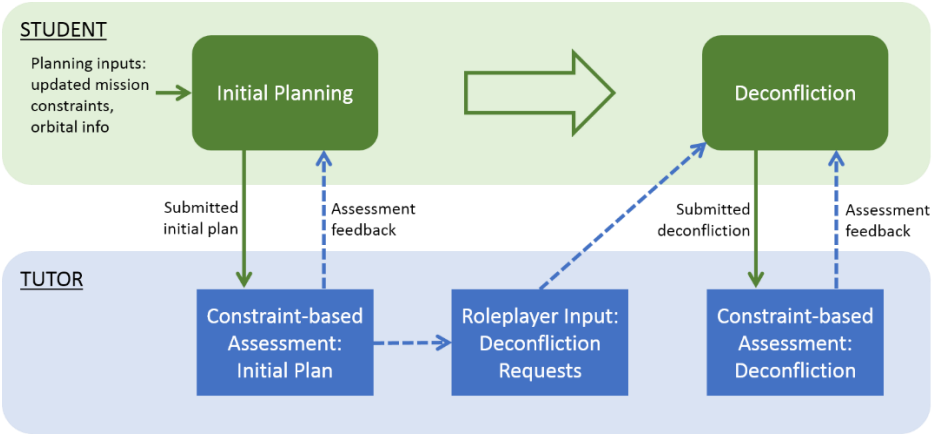

Figure 1. Satellite Planning Exercise Flow

As shown above in Figure 1, each of the planner's tasks has an assessment loop where the student submits a planning artifact (initial plan or deconfliction plan), which is evaluated to produce feedback presented to the student. Plans are only assessed when submitted, and not during the student's progress in plan development, so in this domain the tutor interactions take place in a discrete, turn-based form. There are often many possible variations of planning solutions that satisfy constraints, as opposed to one singular "answer." However, the student must submit a plan that satisfies all constraints before being allowed to proceed to the next exercise step. After the deconfliction task, the exercise concludes and an after action review is presented to the student with a performance summary.

GENERATING CONSTRAINTS FROM DATA

In a training system application, generating constraints from data is the first step in an authoring process that includes other subsequent steps to reach the point of execution-ready assessment, as shown in Figure 2 below.

Figure 2. Assessment Authoring Process Using Derived Constraints

For our test case application, an authoring tool was implemented with user interfaces for all three of these steps. Without elaborating on the subject matter content for constraints, the following are the three main authoring functions.

- Constraint Generation. Constraints are generated in a simple user interface where the author specifies folders containing historical data files, along with basic configuration information such as the satellite involved. This triggers automated processes that parse the data files in their original formats and output constraint files. In the satellite planning application, there are different constraint files for each satellite and each planning task. Different satellites have different mission requirements, and similarly, the planning decisions and data artifacts differ from the initial planning task to the deconfliction task.
- Constraint Annotation. Instructors and authors must be able to review and edit the generated planning constraints that will be used in training exercises, so a user interface is provided with fields for the derived values associated with constraints, where changes may be saved out into new constraint files.
- Scenario Authoring. The final step in the authoring sequence is scenario authoring. In addition to specifying the constraint files that drive how performance assessment will function, authors using the scenario authoring interface to configure other details for an exercise, like the initial inputs the student must use in the planning process, and pre-brief information. In our test case application with a satellite planning simulation-based trainer, the outputs from the scenario authoring utility are directly executable in the trainer.

Constraint files are the conduit for assessment information derived from sample data. The input data are all contained in files directly produced from the operational planning tools in current use, where different planning tasks are associated with different files and formatting conventions. In order to parse each of these kinds of files, initial conversion and integration steps are built into the automated processing of raw source data, first using specialized tools to parse the files in their original formatting from operational tools, and second to link with other necessary information. The data processing step is also where some of the inherent challenges arise. For example, consecutive files in the historical archive may have duplicates or slight time shifts which need to be resolved before running any statistical analysis of the content in a batch processing mode, to avoid problems like counting the same supports twice. After the integration and clean-up tasks, constraints can be derived from the content of the historical solutions.

Constraints are derived using pre-defined guidelines that establish which fields contain the data relevant to individual constraints, and the nature of the data in terms of units, minimum or maximum values, etc. Since constraints are used both in the data analysis process within the authoring tool and also in the runtime assessment mechanisms within the trainer, there is a parallel in how these mechanisms actually function. For data analysis, a set of constraint-specific instructions is needed to determine what information to extract from operational data for the constraint. For performance assessment in training, similar constraint-specific guidelines are needed to establish how the student's decisions (in this case, their submitted plans) satisfy the constraint. Recognizing this parallel, a generalized language for constraint representations was prototyped for this application domain. Within this language, major elements are:

- Scope. Broad category for how a constraint applies. In the satellite planning domain, most constraints are scoped to an individual support, a pair of supports, or a particular time period such as a 24-hour interval.
- Filter. Logical expression that defines additional criteria beyond the scope categories, to filter a narrower subset of supports for which a constraint applies.
- Value. Nested formula using enumerated measurement types, predicates, and attributes.

As an example, the following formula shows how the language is used to specify the Minimum Duration constraint.

Minimum Duration => {Individual: min(attribute(end_time) – attribute(start_time))}

This formula definition conveys that the Minimum Duration constraint is scoped to look at individual support records, and the constraint value is derived from the minimum difference between the *end_time* attribute and *start_time* attribute appearing in the support records. The filter element is not used in this case. Once defined, the same definition can then be used to guide performance assessment with this constraint in a training exercise. This is an initial step toward user-definable constraints, where an author would not be restricted to the constraints already defined, but could define new constraints using this language (or an extended version).

TRAINING APPLICATION WITH GENERATED CONSTRAINTS

In our test case application, the authoring tool was used to generate constraints and construct training scenarios that were executed in a simulation-based satellite planning trainer. Each scenario involves 60 constraints total, derived from source data for three satellites and the two planning tasks (initial planning and deconfliction). Different scenarios and sets of constraints were constructed for experimentation with different subsets of the source data, but a primary "master set" of source data was used to generate constraints for the main evaluation question about the effectiveness of derived constraints in approximating known nominal constraints. Since this paper omits the details and values of the constraints used for satellite planning due to the subject matter, the findings about the correspondence between derived values and nominal values will be discussed in terms of permissiveness, without specific examples and numbers for individual constraints.

Using the constraints derived from the master set of source data, an analysis was conducted by comparing their values against known, nominal constraints gathered from mission requirements documentation. At the first level of analysis, we verified that there were no spurious cases of outliers with nonsensical values among the 60 constraints. This is mostly due to the initial automated step of data conversion and clean-up in the constraint generation process. At the next level, all 60 constraints had derived values within a band of tolerance that was consistent with nominal constraints, but not identical. In the majority of cases, the generated constraints were slightly less permissive than nominal constraints. This is an expected outcome given the nature of performance constraints, although it's worth noting that different elements of performance in different domains may involve harder or softer constraints. In order to illustrate the relationship between permissiveness and hard or soft constraints, the following two examples are constructed to compare a known nominal constraint with the derived value that would come from data analysis alone.

- Soft constraint example: maximum speed limits. Suppose we think of speed limits as a constraint to be derived from "operational" data by sampling actual vehicle speeds on roadways. In most cases, we would likely find distributions that include values above posted speed limits, which would therefore be considered more like soft constraints. So for inherently soft constraints, values derived from data will tend to be *more* permissive than the nominal constraints. For example, if the nominal constraint is a posted limit of 65 MPH, a derived constraint may be a more permissive higher value like 71.3 MPH.
- Hard constraint example: maximum ATM cash withdrawal. Suppose we wanted to derive a constraint value for the maximum single-transaction cash withdrawal from a particular ATM, by sampling actual withdrawal amounts. The distribution would only include values at or less than the maximum, which means any such values used as derived constraints would be *less* permissive than the nominal constraint. For example, if the nominal constraint is a $500 maximum cash withdrawal, the derived constraint might be a less permissive lower value like $400 if there are no data points where people actually withdrew the maximum.

This distinction is mainly useful for interpreting the effectiveness of derived results, but it doesn't need to be a factor in how constraints are represented. The actual mechanisms for deriving constraints from data have the same functionality with either hard or soft constraints. Returning to the application in the satellite planning domain, since most constraints are closer to the hard constraint end of the spectrum, it is expected that the derived values for those are somewhat less permissive.

Performance Assessment Using Derived Constraints

With constraints generated and packaged into scenarios, they can be directly used in the trainer. In the simulation-based trainer, students perform the initial planning task and deconfliction task as shown in Figure 1, each involving a feedback loop where they submit a plan for evaluation and then review automatically generated feedback from the trainer's assessment of their plans. For each task, the assessment mechanism uses the constraints to process the student's submitted plan in two ways (shown in Figure 3 below).

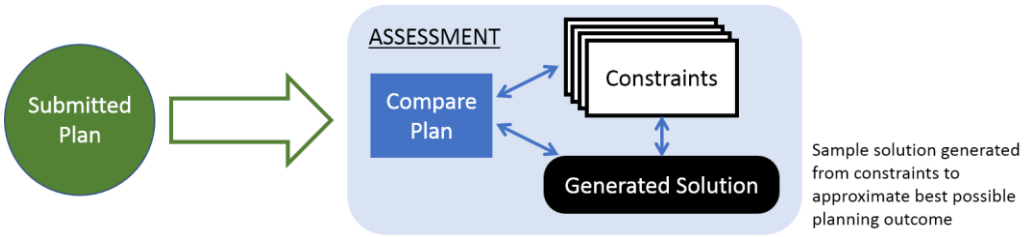

Figure 3. Assessment Mechanism Using Derived Constraints

When the student submits a plan, first it is directly assessed using the constraints. As discussed earlier, the representation for each constraint contains its own information about how to measure satisfaction conditions, which has parallel uses both in the authoring step of processing historical data and in the training runtime assessment step of processing student plans. Using this representation, the assessment identifies any conditions in the student's plan where constraints are not satisfied, and these are collected for feedback to the student. However, a student may submit a solution that minimally satisfies all constraints, while perhaps being unaware if more optimal alternatives are possible. So for more enhanced feedback, a reference sample solution is useful as a form of benchmark for comparison. But it would be undesirable to require human authors to create such a solution for each scenario. As a result, the constraints are used in a generative way as well, to create a nearly optimal solution. This serves two purposes in giving students feedback in planning exercises. First, the assessment feedback can give students a sense for the upper bounds of a possible solution in terms of each constraint. Second, if a student is stuck on a planning exercise and unable to meet all constraints, there is an option to give up on the task and view the generated solution.

Additional Use for Constraints in Roleplayer Agents

During development of the exercise environment, a need arose for a mechanism to simulate roleplayer injections for the deconfliction task. The task flow for satellite planning starts with building the initial plan, for which there are often many possible solutions that can satisfy all mission-related constraints related to a specific satellite. In the operational world, after the initial plan is submitted, a central scheduling authority collects all requests for all satellites into a global aggregated schedule. This global schedule often includes many conflicts in the requests coming from different satellite planners, so a proposed deconflicted schedule is constructed and disseminated back to the planners. In the training setting, the planner's deconfliction task starts with a review of this global deconflicted schedule, to make final decisions about proposed changes on their own satellite requests. So the trainer simulation required a roleplayer agent that can generate a deconflicted schedule that is responsive to the student's initial submitted plan.

We discovered that this was an opportunity to implement a variant of the data-driven constraint authoring approach, applied to behavior modeling. For this application, we used historical data to derive constraints defining a functional model for the kinds of changes made by the central scheduling authority, the relative frequency of different changes, and the content associated with changes. In fact the same set of operational data used to derive performance constraints for the planners' deconfliction task also contained the information needed for the deconfliction roleplayer agents. Since this additional category of constraints was not originally anticipated in the project planning, the mechanism to derive roleplayer agent behavior constraints was only implemented in code, without an accompanying authoring interface. However, it was a beneficial use of operational data that resulted in realistic agent behavior for the training simulation. This also amounts to a potential lesson for other similar applications:

> *Once mechanisms are implemented to mine operational data for constraints intended for performance assessment, there may also be additional categories of constraints that can be similarly derived from the same data sources to meet other needs.*

Initial End User Feedback

Two satellite planners from the operational community participated in an informal evaluation of both the authoring tool and the trainer. Although this is a very small sample, the intention was to gather initial subjective feedback on the first operational version. With expert planners as our initial users, they were essentially evaluating the trainer and authoring tool from the perspective of an instructor or author, as opposed to a prospective trainee. In the first part of the evaluation, end users successfully completed an entire training exercise involving three satellites, which included both planning tasks supported in the trainer (initial planning and deconfliction). This step came first so that planners could become familiar with the training application before beginning to work with the authoring tool, to establish context for the functionality and purpose of the authoring tool. In the next step, they proceeded to create a new training exercise in the authoring tool from scratch. This included generating new performance assessment measures purely from sample data, and incorporating these measures into a new exercise scenario. Finally, the end users performed the new training exercise that they just created. Overall, the end users had favorable comments about both the trainer and authoring tool, and only minor suggestions for user interface changes. The following are some of the aspects that planners cited as noteworthy benefits in the method of authoring from historical data.

- Ability of the authoring tool to use real sample data, directly from archived outputs from operational tools, as the source for derived assessment constraints.
- Additional benefit from the authoring tool's data processing functionality, in highlighting unexpected out-of-threshold cases that appear in historical data. After processing historical data, any unexpected results in the derived threshold values are cause for further investigation into those specific cases and how they occurred in the operational setting. At least one such case was noted during the evaluation.
- Ability of the trainer to perform automated assessment of planning decisions, strictly based on the constraints generated by the authoring tool. Notably, the expert users had some instances where the automated assessment mechanisms identified constraint violations in their exercise sessions, and they welcomed the feedback to improve their submitted plans.

CONSTRAINT GENERATION FOR OTHER TRAINING APPLICATIONS

Our initial test case application with a satellite planning trainer yielded observations that may be helpful for other efforts applying data-driven authoring to other domains. As we encountered unique limitations with the data and workflow associated with satellite planning tasks that essentially placed limits on the functionality of constraint authoring for this domain, we also simultaneously considered how the data-driven approach to authoring might work in more expansive ways for other domains. The following discussion starts with a review of some of the limitations that arose with the application domain, and how the approach might differ for other areas. This is followed by a more general purpose list of domain criteria that can be used to determine if the data-driven approach is more or less feasible for a given training application.

Data Challenges in the Satellite Planning Application

Assessment authoring from other kinds of example sets

The general concept for the initial authoring tool is that annotated examples constitute the data set from which assessment mechanisms are derived. In the satellite planning realm, the examples consisted of a collection of historical solutions, in the form of real-world planning artifacts produced in the operational setting. The benefit of using real operational examples is that they are plentiful, and they truly represent a record of acceptable planning decisions, for both the initial planning and deconfliction tasks. However, historical solutions show good or acceptable performance, but can leave unclear boundaries for bad decisions or performance.

For the authoring tool implemented for the satellite planning application, the constraint annotation interface allows the author to review and edit threshold values associated with constraints derived from the operational examples. But there are other, broader concepts for constraint annotation that did not come into play for this application. For example, in situations where sample data could include student solutions from past exercises, one function of a constraint annotation tool would be to allow instructors to identify cases where constraints are violated in a student's solution and specify what exactly indicates the violation. This annotation activity could take place after an exercise, or even

during an exercise. A reasonable extension would be to accommodate authoring in other contexts, like during training, and with more diverse input data, like a combination of operational examples and training examples.

Under the concept using exercise data as a source for the authoring tool, the system would observe the instructor's interventions over time, and the better it gets at automated assessment, the more the instructor's involvement can be scaled down. This concept relates to methods from the area of programming by demonstration, an active topic of research whose goal is to create intelligent agents that can learn rules of behavior from examples (e.g., Garvey et al., 2009; Castelli, Oblinger & Bergman, 2007; Chen and Weld, 2008). In such a system, the demonstration set should also include examples of incorrect or suboptimal behaviors. The broader application of learning constraints from diverse example sets with positive and negative examples can be constructed as a classifier problem, for which a number of machine learning algorithms are well suited. The end goal is to provide a rich set of annotated demonstrations of both expert and non-expert performance for each scenario. Note that it is still an important component to provide an annotation capability, so that authors can refine the system's derived constraints and assessments.

Looking for hidden patterns in sample solutions

One way that a machine learning approach can add value when deriving constraints is in uncovering hidden patterns in the data. Such patterns can constitute additional constraints to better model the decision space with factors that aren't even captured in explicit nominal mission requirements. In our analyses of the data for satellite planning, we explored several possible avenues for hidden patterns, such as the following example hypothesis question:

> *In the tradeoff space of planning decisions, are there patterns where certain constraints seem to be given greater weight? For example, is there a precedence pattern between any pairs of constraints A and B, such that there <u>are</u> instances where A is met and B is not, but there are <u>no</u> instances where B is met and A is not?*

In this application, no statistically significant pattern was found for this or several other hypotheses that were explored. Yet the prospect exists for such patterns to be present in the data for other applications. Handling such cases is a matter of defining additional constraints using a language such as the generalized constraint representation. In order to look for hidden patterns, the language needs to accommodate self-referential constraints; that is, the additional constraints need to be able to contain references to other constraints to account for the kinds of precedence or correlation relationships described above.

Working with shortcomings in the sample data

There were several shortcomings in the sample data for the satellite planning application. Identifying these factors is useful for considering other applications for the data-driven authoring approach.

- Missing information. At times, satellite planners consider factors that are not reflected in any explicit data artifacts. As the learning algorithm processes example solutions to derive constraints, it cannot discern between decisions that reflect intentional planning preferences vs. outcomes that were forced by hidden factors. For any potential application, there is essentially a threshold where the coverage of decision-making factors in available source data must reach a certain critical mass in order to be an effective use case.
- Operational flexibility vs. nominal requirements. In reviewing the satellite planning constraints derived from operational data, we encountered some cases where operational decisions were not consistent with the nominal stated mission requirements. In further discussions with planners, we learned that some nominal constraints are actually more flexible than others. For example, with one documented requirement on a specific satellite, inspection of the sample data showed that this constraint was only met 71% of the time.
- Changes in mission requirements. Different satellites can have shifting mission requirements over time, which are reflected in the historical planning solutions that may be used to derive constraints. If these changes are not documented, they can lead to unexpected results for the mechanism deriving constraints. As a general rule, this points to the need for cohesive organization of the source data that will be processed. In cases where mission requirements change, constraints should be generated from uniform source data grouped for internal consistency.

Ultimately these challenges from data shortcomings highlight the need for careful data preparation and also the importance of an annotation tool to allow the author to make adjustments to constraints before they are used in training exercises for assessment. This can remedy many problems with input data. It is also important to remember that while operational data can be a useful resource for constraints to be used in training, the training context is still different from the operational context. In some cases where the operational setting allows a degree of flexibility, instructors may not want to teach to a standard of flexibility, electing instead to require students to conform to more fixed nominal constraints. So instructors or authors need to have final say via an annotation tool, about how exactly the constraints should be applied for training. But the data-driven approach to authoring is most effective in applications with clean sample data.

Domain Criteria for Data-Driven Constraint Authoring

Table 1 collects some of the general criteria to consider in determining whether a data-driven approach to constraint authoring is effective or feasible for a given training domain, based on observations from our own test case application.

Table 1. Domain Applicability Criteria

Criterion	Details
Sufficient data	Although it is obvious that sufficient data is needed for a data-driven approach, it bears mentioning that there must be coverage of decision-making factors for the required training tasks. Ideally, positive and negative examples, and minimal missing information.
Human injections captured	Many simulation-based training domains involve significant participation from human instructors, such as injections to add or modify events during an exercise, or roleplayer communications and interactions. If injections factor into student decision-making, they must be captured in data; otherwise the data alone may be prohibitively inadequate.
Discernible performance factors	This factor relates to the nature and content of sample data for reflecting true performance. Example: in a procedural task domain, if every available sample solution contains a sequence where subtask J is performed before subtask K, ideally something in the sample data should indicate whether such sequencing dependencies are required vs. incidental.
Simulation interoperability	There is a presumption that the training simulation supports interoperability with a constraint-based assessment mechanism. The simulation must allow access to internal performance data, and accommodate the presentation of assessment results either internally or in an adjunct user interface for feedback.
Parallels between operational and training data	Data-driven constraint authoring is most effective where there are parallel data structures and processing mechanisms for both operational and training data products. This means that the same mechanisms used to process historical source data to derive constraints can be used to process student actions for constraint-based assessment in runtime.
Synergies and economies of scale	• If roleplayer injections are part of the training tasks, there is the potential to construct automated agent behaviors to produce simulated injections, using the same constraint authoring method with operational data. • Economies of scale can be realized if the training audience involves a community of performers with similarly structured tasks and bounded task variations among different units, or also if there are regular changes in mission requirements and constraints. In such cases, the ability to derive constraints from new operational data sets on a recurring basis is more likely to have benefits over authoring methods that involve hand-coding.

CONCLUSIONS

For the test case application domain of satellite planning, one of the primary goals was to determine if constraints derived from historical data can effectively mimic known constraints, for performance assessment in training. The development process showed that this was achievable with a real-world domain, and produced a number of findings about the approach and its utility for different applications.

- The data-driven approach can provide effective authoring for non-developers. In their first session with the tools, end users from the satellite planning community successfully used the authoring tool to create and run a new exercise with automated performance assessment. With no coding, they started from scratch with collections of sample data in their native source file formats, generated assessment constraints, reviewed and annotated them, and integrated them into a new exercise scenario.
- Effective authoring requires sufficient data. Like all machine learning and data-dependent applications, there is an obvious prerequisite for sufficient "good" data. For any candidate application, this amounts to a judgement call on the tradeoffs between efficiencies from data-driven authoring versus any challenges in terms of coverage, noise, and pre-processing requirements.
- Annotation capabilities are essential. Even with clean data, the statistical analysis from a machine learning component may produce unexpected normative results. Thus we believe any such approach should include practical tools for non-programmer experts to review and modify generated constraints before they are used for assessment in training.
- The value proposition for constraint authoring from data comes from scale and reuse. Since an annotation capability is a necessary component in the authoring tool, a fundamental value question is: how much does it help to automatically populate the annotation tool with constraint values derived from data, versus starting with blank forms that authors must populate themselves? For domains with small numbers of constraints and limited expectation of reuse, there may be limited added value in data-driven authoring. But for applications with economies of scale and opportunities for reuse, mined constraints can be a time saver.
- Automated roleplayers can also be constructed from derived constraints. With the implementation of a mechanism to process source data containing sample solutions, there is the additional possibility to mine the data for other useful information beyond performance constraints. For training domains that involve human injections, there is the potential to derive behavior constraints from sample data, such that automated agents can act as roleplayers, adding similar injections with similar timing and content during training exercises.
- Derived constraints have parallel applications for both training and operations. Once constraints are derived from sample data and used for performance assessment in training, the same constraints have the potential to be used as decision aids in the operational context. In the same way that they can be used to advise the student of flaws in an exercise solution, they can also be used to advise an operator of flaws in an operational solution, and even potentially assist with suggesting more optimal alternatives.

ACKNOWLEDGEMENTS

The concepts described in this paper were developed under a project funded by the Air Force Research Laboratories. The views expressed in this paper are solely those of the authors, and do not necessarily reflect the opinions of sponsors or any Department of Defense agency. The authors are grateful to all Air Force participants who contributed guidance and feedback during this effort. Approved for public release. Cleared, 88PA, Case # MSC/PA-2018-0202 / 88ABW-2018-3122.

REFERENCES

Castelli, V., Oblinger, D., & Bergman, L. (2007). Augmentation-Based Learning combining observations and user edits for Programming-by-Demonstration. *Knowledge-Based Systems*, *20*(6), 575-591.

Chen, J.-H., & Weld, D. S. (2008). Recovering from errors during programming by demonstration. Proceedings from *Proceedings of the 13th International Conference on Intelligent User Interfaces*.

Crowley, R.S., Legowski, E., Medvedeva, O., Tseytlin, E., Roh, E., et al. (2007). Evaluation of an intelligent tutoring system in Pathology: Effects of external representation on performance gains, metacognition, and acceptance. *Journal of the American Medical Informatics Association, 14*(2), 180–190.

Garvey, T., Gervasio, M.T., Lee, T.J., Myers, K.L., Angiolillo, C., Gaston, M.E., et al. (2009). Learning by Demonstration to Support Military Planning and Decision Making. *Proceedings from The Twenty First Conference on the Innovative Applications of Artificial Intelligence*.

Kulik, J.A., & Fletcher, J.D. (2016). Effectiveness of intelligent tutoring systems: a meta-analytic review. *Review of Educational Research, 86*(1), 42-78.

Ma, W., Adesope, O.O., Nesbit, J.C., & Liu, Q. (2014). Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology, 106*(4), 901–918.

McLaren, B.M., Koedinger, K.R., Schneider, M., Harrer, A., & Bollen, L. (2004). Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files. *In the Proceedings of the Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, Seventh International Conference on Intelligent Tutoring Systems (ITS-2004)*. August 2004.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: The constraint-based approach. *IEEE Intelligent Systems*, (4), 38-45.

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

Woolf, B. P. (2008). *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning*. Morgan-Kaufmann.