Automated Scenario Generation: Toward Tailored and Optimized Military Training in Virtual Environments

Alexander Zook, Stephen Lee-Urban, Mark O. Riedl
College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
{a.zook, lee-urban, riedl}@gatech.edu

Heather K. Holden, Robert A. Sottilare, Keith W. Brawner
U.S. Army Research Laboratory
Orlando, Florida USA
{heather.k.holden, robert.sottilare, keith.w.brawner}@us.army.mil

ABSTRACT
Scenario-based training exemplifies the learning-by-doing approach to human performance improvement. In this paper, we enumerate the advantages of incorporating automated scenario generation technologies into the traditional scenario development pipeline. An automated scenario generator is a system that creates training scenarios from scratch, augmenting human authoring to rapidly develop new scenarios, providing a richer diversity of tailored training opportunities, and delivering training scenarios on demand. We introduce a combinatorial optimization approach to scenario generation to deliver the requisite diversity and quality of scenarios while tailoring the scenarios to a particular learner's needs and abilities. We propose a set of evaluation metrics appropriate to scenario generation technologies and present preliminary evidence for the suitability of our approach compared to other scenario generation approaches.

Categories and Subject Descriptors
K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer-assisted instruction (CAI)*. I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*.

General Terms
Algorithms, Design, Theory.

Keywords
Scenario Generation, Military Training, Optimization.

1. INTRODUCTION
Virtual simulations and other game-like virtual worlds are progressively being adopted for "serious" purposes, such as training and education. Computer-based training systems and games share many similarities: both involve a progression of skill-based activities of increasing complexity and difficulty. These activities are often structured through a narrative, mission, quest, or scenario. In this paper we argue for a combinatorial optimization search approach to selecting and ordering events in a training scenario being generated from scratch for use in a virtual training environment. We motivate the need for increased automation of narrative and scenario generation, present desiderata for an automated scenario generation system, describe our particular optimization search approach, and provide an analysis of our approach relative to other possible approaches.

Scenario-based training has individuals or small teams assume the role of an expert for the purposes of practicing skills and knowledge in realistic situations in a *learning-by-doing* approach to performance improvement. The term "scenario" can refer to many different things. In this paper we define a scenario to be a sequence of events that is expected to unfold over a period of time, which if executed will require a trainee to perform a given set of skills. We focus on scenario-based training in virtual environments from a military perspective; the typical use of a scenario in military training contexts is to create the appearance that the trainee is on a mission or is otherwise engaged in an operation in which the given skills will become pertinent within a larger, realistic context. Our approach, however, generalizes to games and other situations where individuals use a set of skills within an authored context—e.g. performing skill-base quests within a larger story arc.

In the military, more training is always desirable, but force generation cycles—the time between deployments that can be devoted to training and improvement—are often short. While scenario-based training is the accepted norm and can be highly effective, it is also costly and inefficient, requiring significant resources: time, space, confederate role players to play oppositional or neutral entities, and scenario developers. Computer-based training, in which scenarios unfold in a virtual environment, can make training more cost-effective and more frequent, especially when using non-player characters (NPCs) to fill non-trainee roles such as opponent forces. However, scenarios must first be written. As long as scenarios are written by human subject matter experts, the fundamental bottleneck to training remains the number, diversity, and appropriateness of scenarios that can be played in computer-based learning environments.

In this paper, we present an approach and preliminary results on a scenario generation system. A scenario generator is a computational system that solves the problem of producing a training scenario—a sequence of events expected to unfold in a training environment—given knowledge about learning objectives, learner attributes, and domain knowledge. Automated scenario generation has several advantages over human-crafted scenario development. (1) Automated scenario generation can *augment* existing human-based scenario development practices to make more training scenarios available, thereby increasing the frequency at which training can occur in virtual environments. (2) Automated scenario generation can *tailor* scenarios to individuals based on their needs and abilities, thereby increasing the

FDG '12, May 29–June 1, 2012, Raleigh, NC, USA.

effectiveness of trainee experiences in the virtual environments. (3) Scenario generation can produce content *on demand* so that learners do not need to wait for human instructional designers to prepare additional training materials.

We model scenario generation as an optimization process: find the sequence of events that maximizes a set of criteria including: the balance of training events of desired types, tailored difficulty, plausible mission context, and the use of affect-building and dramatic events. Specifically we use a genetic algorithm to generate ordered sequences of events (scenarios) from authored domain knowledge regarding scenario events, orderings among groups of events, and the impact of events and groups of events on the quality of a scenario. Unlike other comparable systems that use planning to find the sequence of actions that transitions an initial state to goal situation [4; 8; 9; 10] our approach makes more effective use of the evaluation criteria to choose the right set of scenario events to respond to trainee individual differences, and can create a greater diversity of scenarios. Our system is demonstrated in the military training domain of tactical field care—the decisions about when and how to administer first aid in combat situations.

We believe our approach readily generalizes to other entertainment-based games due to the flexible domain knowledge structures employed and the general way in which player's abilities are modeled. Training scenarios are intimately related to missions and quests within role-playing games, adventure games, first-person shooter games, and so on, where a narrative contextualizes the need to employ increasingly complex skills such as combat or puzzle solving. To the extent that computer games seek to personalize game play and narrative structure, we believe computer games may benefit from our system in similar ways.

This paper provides the following contributions. First, we provide desiderata for automated scenario generation (Section 2). Second, we describe work on a system that solves the problem of automatically generating training scenarios through optimization (Section 4). Third, we present a set of evaluation criteria for scenario generation systems (Section 5) and analyze our work to date.

2. DESIDERATA FOR SCENARIO-BASED TRAINING

Scenario-based live-exercise training is the gold standard for military training, but can also be very expensive. In scenario-based exercises trainees learn to apply knowledge, develop skills, and master concepts under conditions that are representative of the environment in which they will operate in the real world. Scenarios are designed to provide an experience that closely approximates what will occur in real world operations with the added benefit that the events in a scenario are also tied directly to training objectives. Sequences of scenarios are often designed with increasing complexity to gradually improve trainee performance. For example, initial scenarios may focus on well-defined tasks with few of the stressors that are present in the operational environment. Later scenarios may include more complex or ill-defined tasks where decision-making can be exercised and stressors orthogonal to skills compete for attention.

Computer-based training games and simulations complement live training by allowing low-cost use outside of formal training environments. While computer-based training enables more frequent training, it is limited by the number of unique training scenarios available for trainees to practice with. This limitation is compounded when training scenarios are tailored to the specific abilities of a learner. Figure 1 shows the basic scenario development pipeline in which a human scenario developer, using knowledge about learning objectives and the real world, crafts scenarios that can be used in virtual games and simulations. Ideally, the learner's attributes should be changed (positively) due to the learner's exposure to the virtual environment and scenario.

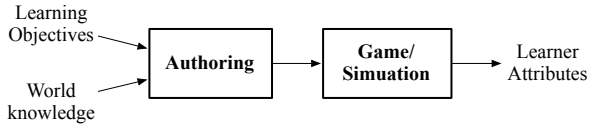

Figure 1. The basic scenario development pipeline.

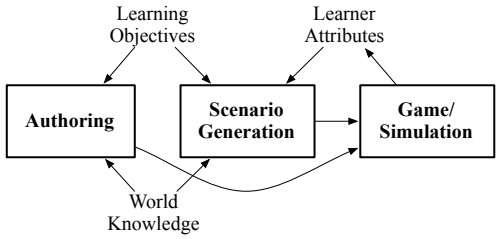

Figure 2. The augmented scenario development pipeline.

Scenarios for computer-based learning environments can be consumed faster than they can be produced resulting in a scenario development bottleneck. Automated scenario generation is one way to overcome the scenario development bottleneck. A scenario generator integrates into the human authoring scenario development pipeline as shown in Figure 2. An automated scenario generator system uses computational representations of learning objectives, world knowledge, and learner attributes to construct tailored scenarios for a learner. The inclusion of learner attributes, in the form of a learner model, is what enables the augmented scenario development pipeline to tailor learning experiences. Note the cycle created between scenario generator, execution environment, and learner attributes: as the learner grows in abilities, the system can adjust its scenarios in lockstep. Learning objectives and world knowledge can be altered to meet changing learning needs or real-world conditions.

We argue that a scalable scenario generation system should have the following capabilities:

- **Replayablility.** Skill mastery is correlated with the ability to generalize across numerous realistic experiences, thus a greater number of scenarios exercising the same skills is desirable [3]. While computer-based training enables more frequent training, it is limited by the number of unique training scenarios available for trainees to practice with. There are rapidly diminishing returns when one trains on the same scenario multiple times—individuals memorize the scenario instead of learning the underlying concepts. To minimize diminishing returns of repeated scenarios, a scenario generation system must be capable of generating numerous distinct variations from a given set of input parameters.
- **Tailoring to individuals.** Scenarios are typically developed to average trainee abilities, delivering sub-optimal experiences for above-average or below-average trainees. Applying the theory of Zone of Proximal Development [16] to any given scenario designed for the average learner, above-average individuals

will experience boredom and below-average individuals will experience frustration. Tailoring scenarios to individuals or small groups can alleviate this skill mismatch and more effectively customize scenarios to the specific needs and abilities of individuals. To achieve this capability, a scenario generation system must incorporate learner differences into the system.

- **Adaptation to new conditions in the world.** Scenarios have a "shelf life," they become less effective as new tactics, techniques, and procedures are adopted to respond to an ever-changing world. A scenario generation system must be capable of being re-configured to account for changes to missions that the scenarios are meant to emulate, new training objectives, and new challenges. This is enabled by easily configurable knowledge structures.

We employ a genetic algorithm to meet these scenario generation criteria, enabling rapid creation of a diversity of scenarios that incorporate author and learner requirements into the generation results. Our flexible knowledge structures enable authors to easily configure new requirements for scenarios or new domain knowledge.

3. RELATED WORK

Computer-based technologies for training and education are used in all aspects of government, industry, and educational institutions. Intelligent Tutoring Systems (ITSs) attempt to computationally replicate the effectiveness of one-on-one human tutoring, making educational remediation more personalized and adaptive to an individual trainee. In STEM (science, technology, engineering, and mathematics) education, Intelligent Tutoring Systems (ITS) have been shown to effectively increase student proficiency by an average of one standard deviation [15]. Game-based intelligent tutors are especially beneficial for the U.S. Military, where training must be highly interactive, immersive, engaging, adaptive, and challenging. Examples of successful game-based tutoring systems in the Military include the Tactical Action Officers trainer [1] and the Tactical Iraqi game-based language and cultural skills learning environment [7]. In STEM education, Crystal Island [12] combines ITS and serious games to teach middle school biology by balancing remediation, problem-solving, and narrative engagement.

VanLehn [14] characterizes tutoring systems as employing outer and inner loops of interaction with a learner. The outer loop selects the next task for the learner to perform based on information about the learner, including traits, learning goals and needs. The inner loop closely monitors every action the learner takes while performing the given task and uses this information to update a model of the learner and provide directed feedback. Scenario-based training under the guidance of an intelligent system can be considered a form of tutoring. A scenario generator functions as outer-loop problem generation. Any performance-based feedback operating during scenario execution could function as the inner-loop remediation. The outer loop process is the primary interest area of this paper.

Hullett and Mateas [5] describe a system that generates training scenarios in a firefighting domain. In this system scenario generation is equivalent to generation of a partially collapsed building that trainees must navigate to rescue victims; the scenario is the environment as opposed to the events and activities that are expected to occur. Our system generates the sequence of events expected to occur as the trainee attempts to perform a role. This allows us to reason about the set of skills that must be performed and dramatic arc, and also to optimize the experience according to the learner's needs. While we do not generate the world the scenario unfolds in, systems such as those demonstrated by Hartsook et al. [2] illustrate methods to incorporate world generation into story generation systems.

A form of experience tailoring can be achieved through an interactive technology called *drama management* (or, *experience management* [10] in the case of serious games). Drama management uses computational techniques to guide a user toward a set of specified objectives by acting through virtual characters to change the virtual world's state. The Automated Story Director [10] uses a planner to generate a branching story structure where all branches lead toward learning objectives. The Adaptive Educational Interactive Narrative System [4] reactively selects the next learning objective and plans forward to it. In both systems, scenario branches are generated in response to physical necessity and are not tailored to the individual. The learner will have repeated experiences if he or she makes similar decisions.

The Interactive Storytelling Architecture for Training (ISAT) [6] also uses interactive narrative to facilitate combat medic training. ISAT *reactively* selects training scenes (particular events such as administering a tourniquet) based on a skill model of the learner. Once a training scene is selected, additional scenes may be sequenced to preserve coherence, although there is otherwise little notion of mission. Both ISAT and our system use a learner model to adapt to the learner's needs. Our approach differs by constructing the full sequence of scenario events ahead of time in order to create more realistic scenarios involving training and non-training events.

Niehaus, Li, and Riedl [8] propose a system that makes small changes to existing, hand-authored scenarios to more closely conform to learner needs. This approach theoretically achieves greater tailoring of training scenarios than hand-authoring alone. However, as a form of case-based planning, it requires a large case-base of hand-authored scenarios to begin with. Further, the system is dependent on having a given initial and goal world states, which may not be practical. Our system employs knowledge structures (see below) that are more easily configured than initial and goals states to meet the ever-changing nature of training domains and trainee knowledge.

Scenario generation shares many similarities with *story generation*, particularly methods that use planning (cf., [9] and [11] for recent work in this area). Planners search for a sequence of operations that transform an initial world state into on in which a goal situation holds. The work by Porteous et al. [9] is especially relevant because it attempts to order several *a priori* given key elements based on a theory of dramatic arc, and then fills in the events between with a planner. Our system *automatically* determines which key learning events should be in the scenario based on a learner model and then arranges those key events along with interstitial events that may have a direct impact on the context under which the key learning events are experienced. Because our system is not based on planning, it has the advantage over planning-based approaches of being able to produce a variety of solutions from the same set of input parameters.

Our approach deviates from the more standard, planning-based approach to scenario and story generation. A planner solves the problem of transforming a world from an initial state to one in which a goal situation holds. While many have argued that planning is a good model for story creation, we observe mismatches between planning algorithms and the objectives of

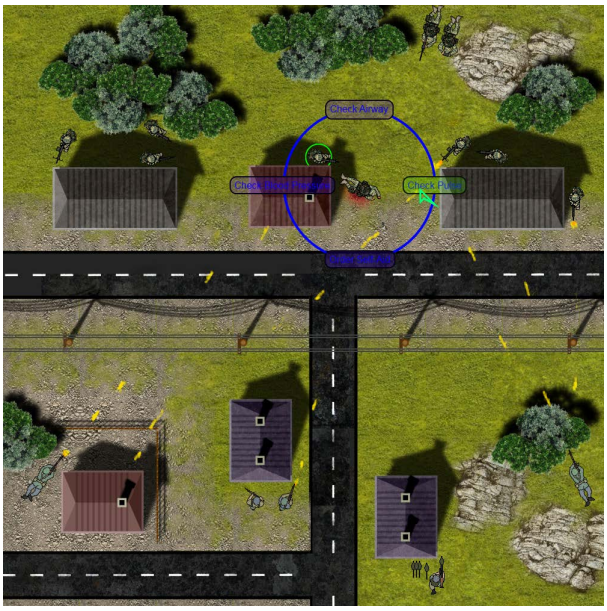

Figure 3. The tactical field care virtual environment. The scenario triggered an ambush in order to wound one of the player's virtual teammates (circled).

tailored scenario generation described in Section 2. First, the primary purpose of a planner is to find a sequence of actions/events that transform the initial state into a goal state. As a formalization of solution correctness, all else is secondary. Beyond correctness, plan optimality is measured by plan length and/or plan operator cost. Such formalizations marginalize the importance of author and learner criteria when developing scenario plans. Second, although search control heuristics can make planning search more efficient, it is challenging to implement complicated planning heuristics for ill-defined domains. A heuristic injects knowledge about which parts of the search space to explore first, but a planner will still terminate if a sound solution is found and none of the heuristics are met. Thus tailoring knowledge or goals may not be met unless explicitly represented. Third, global evaluation criteria require complete event sequences, which are only present at the leaves of a planner's search space, resulting in numerous, expensive sequences of backtracking behavior. Unfortunately, many of the evaluation criteria we propose are global in nature. In contrast, our combinatorial optimization approach sees a complete potential solution on every iteration. Finally, even if optimality assumptions are changed (as in [11]) or violated, a planner still requires an *a priori* known initial state and goal situation. Knowing the initial state and goal state requires the human user to know something about the nature of the mission to be generated. In scenario-based training, the mission is an excuse to practice a certain set of skills; it is thus hard to identify and enumerate relevant concrete initial and goal world states. Poorly chosen initial and goal states might inhibit scenario generation, as in the case where the goal state is a subset of the initial state.

4. OPTIMIZATION-BASED SCENARIO GENERATION

In this section we describe our scenario generation approach. The scenario generator is designed to operate in any domain in which the trainee must practice skills in the context of a larger mission. We demonstrate the system in the context of tactical field care, in which a trainee must make snap decisions under potentially ambiguous situations about whether and how to administer medical care to him/her-self or to others. To provide a realistic context under which decisions about which life-saving skills must be performed, our scenario generator produces a mission reminiscent of actual operations, and embeds tactical field care situations within. Using a learner model, the system can select events to practice skills at the appropriate level of challenge and can also vary the non-skill events to create dilemma situations where it may be more or less clear which decisions to make or skills to perform. Figure 3 shows a screenshot of the virtual environment for tactical field care that we developed to test and demonstrate out scenario generator.

The goal of our scenario generation system is to find the best sequence of events that (a) creates the appearance of a realistic mission, (b) achieves a set of training objectives suitable for a given individual trainee, and (c) is tailored to the individual trainee's abilities. In addition, it is desirable to be able to create a variety of scenarios so that training can be performed multiple times without repetition. Given this problem definition, we base our scenario generation system on combinatorial optimization search. Combinatorial optimization search attempts find one or more structures that maximize a given evaluation function. It is particularly suitable for problems where there are many soft requirements that describe ideal relationships between different aspects of the structure, but few binary requirements such as necessitated goal states. Combinatorial optimization also operates well in domains with a rich variety of potential substructures that constrain and interact with one another but may not be well known *a priori*.

Our scenario generation system specifically employs a genetic algorithm to search for the best solution. A genetic algorithm starts with a population of randomly generated potential solutions and attempts to modify and/or combine aspects of different members of the population to improve the fitness of the population according to the given evaluation function. We detail our scenario generator's knowledge representations and other knowledge sources, evaluation function, and algorithm in the subsequent sections.

4.1 Scenario and Knowledge Representation

The scenario generator is responsible for creating the set of events that comprise a training scenario. We model a scenario as a totally ordered (i.e. linear sequence) series of story events. An event is a primitive element that describes a small number of actions. Example events include various ways in which characters can engage in combat, treat injuries, engage in non-combat related mission activities (e.g., patrol), and socialize with other characters in the mission. A fragment of a tactical field care scenario is shown in Figure 4. Note the use of social events such as build-affinity and bullied to set up a dilemma situation where the learner must choose who will receive medical care and then apply the appropriate procedure.

There are several types of knowledge required to generate such a scenario: an event template library, required events, ordering constraints, and evaluation grammar. The *event template library* contains the set of all possible events that can occur in a scenario. Each event is defined by an event type and a set of parameters that determine the aspects of the event including: the characters participating in the event, impact of the event on the affinity of characters for one another, types of injury sustained by characters in the event, context of the training (while the player is under fire or after combat), and type of the event (socializing, engaging in

```
…
Presence-patrol (market)
Make-friends (private)
Bullied (sergeant)
Search-house (…)
…
Ambush (…)
Get-shot (private, severe-leg-wound)
Get-shot (sergeant, sucking-chest-wound)
Enemy-retreat (…)
Give-care (sergeant, chest-compression)
Get-thanked (sergeant)
Die (private)
Medical-evac (…)
…
```

Figure 4. Fragment of a tactical field care scenario.

combat, or mission-specific). It also describes parameterized preconditions and postconditions of each event type used only in the post-processing stage. During generation, the system may select and instantiate event templates into the scenario. This entails selecting a template and filling it with appropriate values given for each event slot. Events in which skills are practiced, called *skill events*, are annotated so that the learner model can determine how they will impact the particular trainee when instantiated into a scenario.

Required events are events that must be in the scenario for it to achieve necessary learning objectives. For example, if we wish a learner to practice deciding when and how to treat a leg wound on a teammate after the team is safe from fire, then required events can be used to force an event of *treat-leg-wound* to be present in the scenario. Required events can be partially-instantiated, meaning that certain optional parameters, such as *who* needs the care can be discovered opportunistically later. Required events can enforce learning objectives as the example above, but can also place requirements on the nature of the mission the trainee will experience (e.g., presence-patrol) or the types of danger encountered (e.g., an improvised explosive attack).

Ordering constraints may be specified in advance to provide hard rules that particular sub-sequences of events must occur in a particular order. For example, a scenario may be required to ensure that all events of applying the skill of administering a tourniquet to the leg are preceded by a teammate sustaining a leg wound injury: *get-shot*(c, t, i=leg-wound) < *give-care*(c, s=tourniquet)

The *evaluation grammar* specifies a set of soft ordering constraints and hierarchical compositions of ordering constraints to make graded assessments of the structure of a scenario. Constraints in the evaluation grammar encode dramatic or pedagogical preferences for training scenarios used during scenario evaluation. The grammar informs the system of familiar subsequences that it might not otherwise recognize as favorable, such as common ways in which a presence patrol mission unfolds or common social patterns. A scenario in which most events are matched against the evaluation grammar is considered superior to scenarios in which fewer events are explained by the grammar. The genetic algorithm thus favors the use of more canonical sequences. However, since the grammar imposes soft constraints, the generator is able to violate the stereotypical situations in order to maximize other factors and increase the overall quality of the scenario. Figure 5 shows a fragment of evaluation grammar. Rules can be sequences or "OR" clauses. Italics denote abstract symbols while non-italics denote events expected to be instantiated in a scenario. Event parameters are tracked by the grammar when necessary to enforce consistent parameter usage (e.g., characters) across many events. The fragment of grammar in Figure 5 specifies that a good pattern of events includes the player building affinity for a particular character (e.g., make-friends) and then having that character become wounded. Likewise, a common social convention has a character give thanks for medical attention.

```
…
dilemma → pity OR single-save OR keep-cover
social → ct OR bl
pity → bsc OR asc
bsc → build-aff(c), get-shot(c), give-care(c)
single-save → build-aff(c1), lose-aff(c2), get-shot(c1),
                get-shot(c2), give-care(c2), die(c1)
ct → give-care(c), get-thanked(c)
…
```

Figure 5. Fragment of the evaluation grammar.

4.2 Evaluation Functions

A scenario is evaluated using a set of functions that examine the global sequence of scenario events to determine its overall quality. Our system currently employs a set of encoded functions to assess developments of the dramatic arc and inclusion of training skills. The current set of functions includes:

- **Character use:** to avoid character proliferation, scenarios that maintain a small cast of repeatedly used characters are scored more highly.
- **Event type use:** scenarios that employ a variety of types of events are scored more highly.
- **Event type flow:** grouping the types of events used over the course of the scenario (e.g., having clustered sequences of combat events or social events) is scored more highly [17].
- **Affinity use:** a running affinity level score calculated over the course of events compared to various combat situations characters are involved in; injuring characters that have built affinity with the player beforehand is scored highly.
- **Scenario length:** scenarios closer to the specified duration are scored more highly.
- **Evaluation grammar:** the structure of preferred event orderings; more events accounted for by higher-level rules are scored higher.
- **Learner model:** to tailor the training scenario, we use a learner model to reason about how the learner will respond to skill events in the scenario. The learner model maps skill events (and other features such as affinity) to a performance value. A performance curve is computed for the scenario and compared to a given, target curve (see Figure 6).

All of these functions take parameters provided by an author specifying the thresholds used for assessment (e.g., target difficulty curve, target scenario length, etc.) and relative weighting of the features. A scenario's fitness is determined by the weighted linear sum of all evaluation functions.

Our combinatorial optimization search approach to scenario generation is domain-independent in the sense that the evaluation critiera can be swapped out for different criteria, although we believe our current set is relatively general. A majority of domain specific information is captured in the evaluation grammar, which can be easily re-authored.

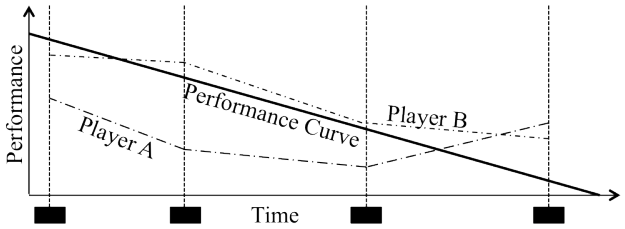

Figure 6. Illustration of the learner model evaluation. Due to individual differences, two individuals are shown to respond to the skill events (black boxes) differently.

The learner model evaluation function is the primary way in which scenarios are tailored to an individual. The learner model evaluation function identifies all the skill events in the scenario and evokes a learner model to predict the learner's performance on the skill events. The prediction is compared to an author-specified target curve indicating a desired trajectory of learner performance over the scenario. Figure 6 depicts a monotonically decreasing target performance that may convey the sense of increasing difficulty. Other curves may be authored to create different learner experiences.

Note that our learner model captures learner *performance*, not scenario *difficulty*. Learner performance is directly observable and can be evaluated according to pedagogical criteria, whereas scenario difficulty is a subjective experience related to performance but does not necessarily correspond to performance levels or learning. Many factors outside of performance may impact the feeling of difficulty, including task ambiguity, learner fatigue, or a learner's prior knowledge. To avoid these complications we employ a performance measure defined by the domain author that scores learner actions on skill events according to pedagogical criteria. As an example, performance scores for treating a leg-wound may be: high if a tourniquette is used within a short time window, moderatate if a tourniquette is used after a minor delay, and low if the tourniquette is not used. Acquisition of this learner model data is beyond the scope of this paper.

4.3 Scenario Generation Process

Our scenario generation process employs a genetic algorithm. Genetic algorithms are an optimization technique commonly used in domains where quality of resulting products can be easily assessed but methods for describing the process of generating these products are poorly understood or time-consuming. Genetic algorithms are defined by a method to evaluate candidate products (here scenarios) along with a fixed set of operations to change particular elements of the product. A genetic algorithm (GA) proceeds by generating a pool of candidate solutions (called the population). The GA iteratively alters and evaluates the members of this population, typically keeping a subset of high-quality solutions and discarding and replacing low-quality solutions. The GA halts when it achieves a desired number of solutions of a desired quality.

The process begins by reading in author-specified domain knowledge that details the types of events possible and required in the scenario, ordering constraints on those events, and specifications for evaluation functions. An initial population of potential scenarios is generated including all required events and a random subset of optional events, with the required orderings on these events enforced. The iterative process performs one of four possible alterations to the scenarios over each step:

- **Addition:** Instantiates an event template and inserts it into the scenario at a random location.
- **Deletion:** Removes a random event from the scenario.
- **Mutation:** Randomly alters a parameter of a randomly chosen event in the scenario.
- **Cross-over:** Given a pair of scenarios in the population, randomly swap the first *n* events in one scenario with the first *m* events of the other scenario.

Evaluation of the scenario employs the fitness function described in the previous section to assess the quality of each member of the population. The top 10% of scenarios are retained unchanged, the bottom 10% are discarded and replaced with newly generated scenarios, with the remaining scenarios altered across iterations. The alteration and evaluation cycle repeats until the maximum fitness among the population makes no substantial gains for 100 iterations.

4.4 Post-Processing Stage

Although we argue that combinatorial optimization search is the best algorithmic fit for scenario generation, planning has one key advantage over combinatorial optimization search: planners can easily guarantee the binary property of causal coherence. Causal coherence is the extent to which any event is necessitated by prior events [13]. Global binary constraints such as causal coherence are difficult and expensive to achieve using combinatorial optimization search as they introduce sharp discontinuities in the space of scenario qualities. Consider ensuring consistent movement between locations: the trainee's avatar could be on a presence patrol in part of a city and then suddenly be shot and wounded in another part of the city with no explanation for how he got there. Solving such inconsistencies adds little to the quality of the solution relative to the increase in search time for combinatorial optimization. At the end of the optimization stage, we have a sequence of events with undetermined causal coherence. To resolve coherence issues, our system uses partial-order planner as a post-processing stage to ensure the proper causal linkages exist between the events selected for the scenario.

The partial-order planner solves the problem of establishing the preconditions of all events in the sequence. The preconditions of all events in the scenario so far are posted as goals to the planner. During the process, new events may be instantiated into the scenario that link events to earlier events and resolve any causal inconsistencies. Most events inserted during this post-processing stage involve (a) location changes of the NPCs to establish coherence of events and to create and position enemy forces, and (b) ensuring entities possess the requisite resources to participate certain events. Because there is no initial world state, certain special events that create entities in the virtual environment—such as *create-enemy-force*—do not have any preconditions. These special actions, first proposed by Niehaus, Li, and Riedl [8], can be ordered prior to the first event in the scenario to essentially create the initial world state required by the optimal scenario.

5. SYSTEM EVALUATION

A scenario generator aims to provide a variety of tailored scenarios that meet the training needs of an individual. We propose the following metrics for success of a scenario generation system: (1) the quality of solutions as a function of running time, (2) the diversity of scenarios as a function of running time, and (3) the performance of a trainee and appropriateness of difficulty level when training on generated scenarios. In this section we apply the first two evaluation criteria to our optimization-based

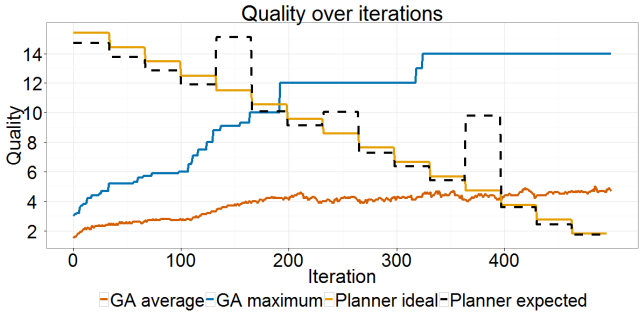

Figure 7: Population quality across iterations.

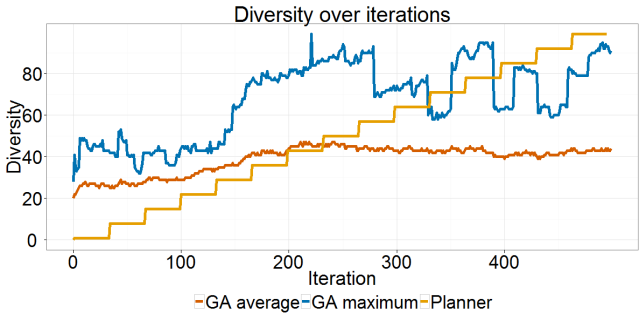

Figure 8: Population diversity across iterations.

scenario generation system, and compare it to a hypothetical planning-based scenario generation system. We have not yet run studies of human skill learning with and without the scenario generation system.

Scenario quality measures the extent to which the scenario meets the given success criteria. Pedagogical criteria include the extent to which the scenario meets a given set of learning objectives (i.e., does it train the right things?) and have the desired difficulty progression for an individual learner. Dramatic criteria include the extent to which the scenario creates a plausible mission context for the desired skills to be trained in. Our system applies an evaluation function (see Section 4.2) to every potential solution. Each iteration of the genetic algorithm produces a population of potential solutions, allowing us to easily measure the quality of the best individual in the population as well as the average quality of the population. Figure **7** (blue and red lines) illustrates the maximum and average fitness levels of our scenario generator over the course of 500 iterations using 100 individuals. The figure shows that the best result is achieved in a step-wise fashion; the generator rapidly achieves a high-quality solution initially then makes occasional, incremental improvements on this solution. Because this evaluation of quality uses the evaluation function employed in generation, an independent assessment of solution quality should also be conducted.

Scenario diversity is a measurement of variations among the content of scenarios. An effective scenario generator should produce many scenario options from the same set of input parameters. The scenarios generated for a given input specification should have substantial differences from each other in order to provide varied contexts in which a learner can practice skills. As with quality, assessing a scenario generator should explore the diversity among scenarios within a population as the scenario is provided greater time and/or memory for results.

We operationalize scenario diversity using the Levenshtein edit distance. The edit distance is commonly used to capture the number of additions, deletions, or substitutions necessary to transform one string of characters into another. We implement the edit distance measure as the number of event changes in terms of symbol (e.g., changing "get-shot" to "give-care") and parameters (e.g., changing "private" to "sergeant") required to make two scenarios identical. This captures basic differences between events at the level of both the type of event and its content. Richer metrics for scenario diversity may exist that incorporate semantic or ontological knowledge of event parameter similarity, although any unbiased diversity metric will suffice for comparison between systems.

To evaluate our system's ability to create diverse solutions, we compute the edit distance between all scenarios in the population. Both the average and maximum values across the population may be extracted. Figure 8 (blue and red lines) illustrates these values per iteration. These results indicate the generator gradually increases the diversity among population results.

How does our approach compare to planning-based scenario generation approaches? Unfortunately a direct comparison is difficult or impossible because different systems require different input parameters, solve slightly different problems (i.e., have different definitions of scenario), and work for different domains. Ignoring differences such as how knowledge is represented and initial and goal states, we make the following general observations about how our system would compare to a typical planning-based system. Our suggested metrics—quality over time and diversity over time—require a time variable. Therefore we make the assumption that the hypothetical planning-based scenario generator can be run some number of times, retaining the search space from run to run so that it can generate *n* plans. We further assume the planner finds a complete plan every *m* iterations. Thus the playing field has been leveled in the sense that both the genetic algorithm and planner have a number of iterations.

The orange line in Figure 8 shows the theoretical performance of a planner that generates *n* plans from the same search space, assuming a plan is found every *m* iterations. At first, there is only one plan, so diversity cannot be measured. Note that the anticipated behavior of the planner as described is a steady increase in diversity. We expect any given complete plan found to be "near" the most recent solution in the search space, resulting in small steps in diversity but a steady march toward greater diversity. This is opposed to the large initial diversity followed by steady increase that we observe in our approach.

For complex, real-world scenario generation domains a planner will require a greater amount of time to generate a diversity of scenarios compared to a combinatorial optimization approach. In large and ill-defined domains this may require substantial time costs as the planner will more exhaustively search single local regions before switching between them. Combinatorial, nondeterministic and parallel approaches are able to explore multiple potential solutions simultaneously, generating greater initial diversity and more rapidly exploring multiple possible tailored scenarios that meet given learning goals.

We expect the opposite behavior from a planner with regard to quality. Assuming the planner's heuristic effectively guides the algorithm to the best solution first, a planner's first plan should have the highest quality. Subsequent plans are expected to be of progressively lower quality as the planner is forced to continue finding plans against the direction of the heuristic function. In

Figure 7, the orange line describes the theoretical ideal behavior of the planner with respect to quality. Note that in practice, when planners are applied to ill-defined domains where heuristics are typically not known to be admissible one may find potential increases in quality as the space is more fully explored. The black dotted line in Figure 7 shows this phenomenon, which contrasts with our approach that starts out with a population of low quality solutions and marches toward a population of higher quality solutions as iterations increase.

In practice this means a planner would be expected to yield a high-quality tailored scenario early, and produce several scenarios of roughly equivalent quality that are very similar. A combinatorial optimization process is more likely to provide lower-quality solutions initially, but to converge onto multiple different regions containing high-quality solutions. As such, we expect the combinatorial approach to iteratively refine multiple distinct scenarios that meet provided learning objectives, rather than explore variations on the same high-quality scenario.

6. CONCLUSIONS

In this paper we argue for combinatorial optimization search as the means for selecting and ordering events in a training scenario being generated from scratch. This approach has the potential to provide the greatest amount of diversity in solutions for a given set of input parameters. Further, this approach allows for the evaluation of the total course of events rather than building the scenario one event at a time. It easily incorporates a learner model, providing the opportunity to tailor the scenario, and provides a flexible means of providing knowledge about scenario structure, dilemma, and other training domain considerations. Preliminary analysis shows that our approach is capable of generating a rich diversity of scenarios over time, and also optimizes scenario quality over time.

We envision this system extending to entertainment-based games that incorporate skill-based challenges and aesthetic narrative content, enabling games tailored to players while relieving constraints on game authors in developing a set of possible game narratives. This framework is particularly effective for games with decomposable parts of narrative content, such as quest-based story structures in role-playing games.

As one adopts virtual technologies for training and games, the need for scenarios increases dramatically. In addition, tailoring of the scenario and learning environment to the needs and abilities of an individual learner can lead to more effective, on demand training opportunities. This requires augmenting the traditional human-based scenario development pipeline with a closed-loop scenario generation systems that incorporates learner attributes and theoretically leads to more effective training as learners have greater opportunities to train on more relevant scenarios.

7. ACKNOWLEDGMENTS

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

8. REFERENCES

[1] Craighead, J. 2008. Distributed, game-based, intelligent tutoring systems—the next step in computer based training? *Proceedings of the International Symposium on Collaborative Technologies and Systems*.

[2] Hartsook, K., Zook, A., Das, S., & Riedl, M.O. 2011. Toward Supporting Stories with Procedurally Generated Game Worlds. Proceedings of the 2011 IEEE Conference on Computational Intelligence in Games.

[3] Hedlund, J., Antonakis, J. & Sternberg, R. 2003. Tacit Knowledge and Practical Intelligence: Understanding the Lessons of Experience. ARI Research Note 2003-04.

[4] Hodhod, R., Cairns, P. and Kudenko, D. 2011. Innovative Integrated Architecture for Educational Games: Challenges and Merits. In *Transactions on Edutainment V*. Springer.

[5] Hullett, K. & Mateas, M. 2009. Scenario generation for emergency rescue training games. *Proceedings of the 4th International Conference on the Foundations of Digital Games*.

[6] Magerko, B., Stensrud, B., and Holt, L. 2006. Bringing the schoolhouse inside the box – A tool for engaging, individualized training. *Proceedings of the 25th Army Science Conference*.

[7] Mills, C. & Dalgarno, B. 2007. A conceptual model for game-based intelligent tutoring system. *Proceedings of the 2007 Australasian Society for Computers in Learning in Tertiary Education*.

[8] Niehaus, J., Li, B. and Riedl, M.O. 2011. Automated scenario adaptation in support of intelligent tutoring systems. Proceedings of the 24th Conference of the Florida Artificial Intelligence Research Society.

[9] Porteous, J., Teutenberg, J., Pizzi, D., & Cavazza, M. 2011. Visual programming of plan dynamics using constraints and landmarks. *Proceedings of the 21st International Conference on Automated Planning and Scheduling*.

[10] Riedl, M.O. Stern, A., Dini, D., & Alderman, J. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. International Transactions on Systems Science and Applications, 3(1).

[11] Riedl, M.O and Young, R.M. 2010. Narrative Planning: Balancing Plot and Character. Journal of Artificial Intelligence Research, 39, 217-267.

[12] Rowe, J.P., Shores, L.R., Mott, B.W., & Lester, J.C. 2011. Integrating Learning, Problem Solving, and Engagement in Narrative-Centered Learning Environments. *International Journal of Artificial Intelligence in Education*, in press.

[13] Trabasso, T. and van den Broek, P. 1985. Causal thinking and the representation of narrative events. Journal of Memory and Language, 24, 612-630.

[14] VanLehn, K. 2006. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education* 16.

[15] Verdú, E., Regueras, L.M., Verdú, M. J., De Castro, J.P., & Pérez, M.A. 2008. Is adaptive learning effective? A review of the research. In L. Qing, S. Y. Chen, A. Xu, & M. Li (Eds.) *Proceedings of the 7th International Conference on Applied Computer & Applied Computational Science*.

[16] Vygotsky, L. 1978. *Mind and Society: The Development of Higher Psychological Processes*. Harvard University Press.

[17] Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Thesis, Carnegie Mellon University.