Automated Case Generation Using a Genetic Algorithm

Hayley Borck
Adventium Labs
111 Third Ave S., Suite 100
Minneapolis, Minnesota 55401
hayley.borck@adventiumlabs.com

Mark Boddy
Adventium Labs
111 Third Ave S., Suite 100
Minneapolis, Minnesota 55401
mark.boddy@adventiumlabs.com

ABSTRACT

Case-Based Reasoning is a learn-by-experience approach in which past problem solving instances, called cases, are used to solve novel input problems. Authoring these cases is often a manual process requiring the assistance of a domain expert. To alleviate this problem, we have developed CBGen, a Genetic Algorithm-based approach for case creation. CBGen uses individuals that represent the initial parameters of a low fidelity simulator and a target task that must be simulated. The encoding of each individual is used to run the simulations and store how the task was completed. Individuals are then evaluated based on the expected benefit of the case they generated being retained by the system. A preliminary proof-of-concept in an augmented reality domain validate the feasibility of using CBGen to automatically create a case base.

CCS CONCEPTS

•Information systems → Expert systems; •Computing methodologies → Heuristic function construction; •Applied computing → Computer-assisted instruction;

KEYWORDS

Case-based reasoning, Hybrid Systems, Genetic Algorithm

ACM Reference format:
Hayley Borck and Mark Boddy. 2017. Automated Case Generation Using a Genetic Algorithm. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3075603

1 INTRODUCTION

Case-based Reasoning (CBR) is motivated by the idea that similar problems tend to have similar solutions. Problem-solution pairs, called *cases,* encode the knowledge that a CBR systems uses to solve novel input problems. However, creating cases can be a knowledge intensive process and, in many domains, case acquisition can be an expensive and time consuming process. Instead of relying on domain experts to manually author cases, we propose an automated approach for case acquisition. Our system, *CBGen*, uses Genetic Algorithms to automatically generate cases that are both novel and allow the CBR system to solve problems it previously could not.

GECCO '17 Companion, Berlin, Germany
 978-1-4503-4939-0/17/07…$15.00
DOI: http://dx.doi.org/10.1145/3067695.3075603

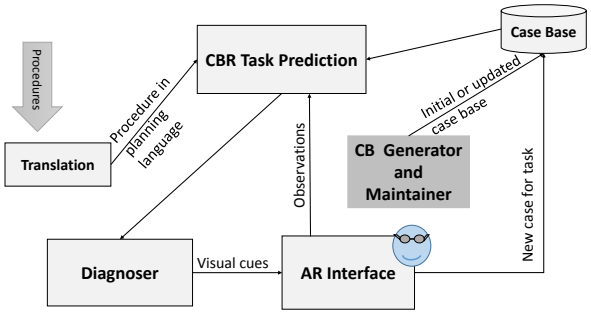

Figure 1: The extended MonitAR system architecture.

In this paper, we examine automatic case generation in the domain of an Augmented Reality Intelligent Procedure Guidance System (ARIPGS). An ARIPGS is an automated tool that guides users through training scenarios where the user must perform a predefined *procedure*. Each procedure is composed on a number of steps, called *tasks*, that must be performed to successfully complete the entire procedure. The ARIPGS is responsible for identifying if the user is having difficulty completing a procedure and assisting them by displaying visual cues on smart glasses that the user wears (i.e., telling them what the next task to complete is). In the ARIPGS context, the *problem* portion of a case is the user's observed behavior, and the *solution* is the task that system should recommend the user completes. Human-authored cases require a deep understanding of the errors a trainee may make and are therefore difficult to obtain.

Our work uses an existing case-based ARIPGS system, MonitAR [2] (Figure 1), with extended capabilities to automatically generate cases (the *CB Generator and Maintainer* module). In MonitAR, the *CBR Task Prediction* module continuously observes the user (through the *AR Interface*) and uses a similarity calculation to retrieve the most similar stored case (the case with the most similar set of observations). The solution portion of the most similar case is given as output to the *Diagnoser*, which may provide visual clues to the user.

2 CBGEN

The individuals in our GA encode starting parameters of a simulation and a task to be performed during the simulation. In our current implementation, the simulation parameters include the user's initial hand position and their handedness (i.e., left or right), and the task includes the action, the target object, and a secondary object (possibly none). For example, if the task is *"pick up the pencil"*, the action is *"pick up"*, the target object is *"the pencil"*, and there

is no secondary object. When an individual's encoding is used to run a simulation (i.e., in place of the *AR Interface*), a simulated user performs the task and a set of observations of its actions is collected. Each such observation-task pair is stored as a case.

The initial population is randomly initialized, and crossover and mutation operators are used. Each operation is restricted to selecting values from a pre-defined set (e.g., handedness, action, object types) or within a pre-defined range (e.g., hand position). Although these values are currently hand-encoded, future work will examine manually extracting them from a domain definition. The initial population is biased such that there is at least one individual for each *known task* (i.e., a task that is necessary for at least one known procedure). Additionally, a set of random individuals are also used. This initialization procedure is used to ensure a diverse set of individuals that encode both correct behavior (i.e., performing known tasks) and erroneous behavior (i.e., performing other tasks). In domains with anticipated errors and failures by users, the ability to generate such observations is invaluable. We use a fitness proportionate selection routine to create one additional individual in the population during crossover. Each individual has a 5% chance of mutation. Additionally, a set of semantic rules are used to handle changes to interdependant features. For example, if the handedness is changed the initial hand position is also changed to a position on the other side of the body.

The most similar case acquisition strategy to CBGen is Automatic Case Elicitation (ACE) [4]. When ACE encounters a novel problem, it randomly generates a solution and later evaluates the quality of the solution. This differs from our own work in that we are generating input problems for known solutions and are generating the unknown information through simulation, rather than randomly. Automatic case generation has also been explored using text extraction [1, 6], but those approaches require a prior knowledge source (e.g., a report) to mine.

3 CALCULATING BENEFIT

In order to evaluate the quality of an individual, the *benefit* [7] of the case it creates is measured. The benefit of a case is the additional *coverage* gained by adding the case to the set of existing cases, called the *case base*. The coverage of a case x is defined as the "set of target problems that it can be used to solve" [5], and that set $N(x)$ is called the case's *neighborhood*. To determine if x can solve another case y, we calculate if their problems (i.e., observations) have a similarity greater than a threshold λ and they have an identical solution (i.e., task). The coverage of x can then be calculated as:

$$Coverage(x) = \sum_{y \in N(x)} P(y) \tag{1}$$

$P(y)$ is a weighting that represents the importance of solving y. In our prototype, all weights are equal (i.e., the coverage is simply the size of $N(x)$). For a set of cases X, the coverage is:

$$Coverage(X) = \sum_{y \in \bigcup_{m \in X} N(m)} P(y) \tag{2}$$

The benefit of adding case x to the existing case base Z is:

$$Benefit(x) = Coverage(Z \cup x) - Coverage(Z) \tag{3}$$

If adding x to Z results in more problems being solvable, the $Benefit(x)$ will be greater than zero.

At the end of each generation, all cases with a benefit greater than zero are added to the case base. Thus, Z is continuously updated as novel cases are evolved using the GA and each newly added case extends the coverage of Z. Once a case w is added to Z, in future generations w will no longer have any benefit (i.e., an exact copy of it already exists in Z so it cannot add additional benefit). This property results in the population of the GA changing rapidly between generations and generating novel individuals. We use Monte Carlo estimation to determine when to stop adding cases to the case base [3]. The Monte Carlo coverage estimation uniformly samples n cases from the problem space (where n ¿ 25% and n ¡ 75% of the population), and measures what percentage of those cases are solved using cases in Z. We stop under two different conditions: (a) when the estimated coverage of Z is greater than 80% of the problem space, or (b) the estimated coverage has remained constant for 3 iterations. In future work, we plan to use a more sophisticated stopping criteria based on the expected benefit of adding additional cases.

We ran an initial proof-of-concept to examine the feasibility of using CBGen to automatically generate cases for use in MonitAR. Our initial validation used a set of 14 types of tasks and 17 unique objects. The case base was initially empty, and after trail and error an initial population size of 1000. All other parameters were as described in this paper. The genetic algorithm converged after 10 generations.

4 CONCLUSIONS

In this paper we presented CBGen, an algorithm to automatically create cases using a genetic algorithm. A prototype of the algorithm was designed for integration with an existing case-based augmented reality training tool, MonitAR, and showed promising initial results. In addition to fully integrating CBGen with MonitAR and evaluating the automatic case generation capabilities, we plan to empirically demonstrate the benefit of automatically generating high-quality cases rather than relying on a domain expert to author them.

REFERENCES

[1] Stella Asiimwe, Susan Craw, Bruce Taylor, and Nirmalie Wiratunga. 2007. Case authoring: from textual reports to knowledge-rich cases. In *Proceedings of the 7th International Conference on Case-Based Reasoning*. Springer, 179–193.

[2] Hayley Borck, Steven Johnston, Mary Southern, and Mark Boddy. 2016. Exploiting Time Series Data for Task Prediction and Diagnosis in an Intelligent Guidance System. In *Proceedings of the 24rd International Conference on Case-Based Reasoning Workshops*. 132–141.

[3] David Leake and Mark Wilson. 2011. How many cases do you need? assessing and predicting case-base coverage. In *Proceedings of the 19th International Conference on Case-Based Reasoning*. Springer, 92–106.

[4] Jay H. Powell, Brandon M. Hauff, and John D. Hastings. 2005. Evaluating the Effectiveness of Exploration and Accumulated Experience in Automatic Case Elicitation. In *Proceedings of the 6th International Conference on Case-Based Reasoning*. Springer, 397–407.

[5] Barry Smyth and Mark T Keane. 1995. Remembering to forget. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Citeseer, 377–382.

[6] Chunsheng Yang, Benoit Farley, and Bob Orchard. 2008. Automated case creation and management for diagnostic CBR systems. *Applied Intelligence* 28, 1 (2008), 17–28.

[7] Jun Zhu and Qiang Yang. 1999. Remembering to add: competence-preserving case-addition policies for case-base maintenance. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. 234–241.