Using GP to Model Contextual Human Behavior – Competitive with Human Modeling Performance

Hans Fernlund[1] and Avelino J. Gonzalez[2]

Intelligent Systems Laboratory
University of Central Florida
Orlando, FL 32816-2450
[1]hfe@du.se, [2]gonzalez@pegasus.cc.ucf.edu

Abstract. To create a realistic environment, some simulations require simulated agents with human behavior pattern. Creating such agents with realistic behavior can be a tedious and time consuming work. This paper describes a new approach that automatically builds human behavior models for simulated agents by observing human performance. With an automatic tool that builds human behavioral agents, the development cost and effort could be dramatically reduced. This research synergistically combines Context-Based Reasoning (CxBR), a paradigm especially developed to model tactical human performance within simulated agents, with the Genetic Programming machine learning algorithm able to construct the behavior knowledge in accordance to the CxBR paradigm. This synergistic combination of AI methodologies has resulted in a new algorithm that automatically builds simulated agents with human behavior. This algorithm was exhaustively tested with five different simulated agents created by observing the performance of five humans driving an automobile simulator. The agents show, not only the capabilities to automatically learn and generalize the behavior of the human observed, but they also exhibited a performance that was at least as good as that of agents developed manually by a knowledge engineer.

Introduction

Building human behavior models can be very complex and time-consuming. Extracting and processing the knowledge from the subject matter expert (SME) is a very intricate task. To get the expert to express the behavior in an articulate way, analyze the information and later implement it in the model also clear sources of misinterpretation. It is almost impossible to develop a mathematical formalism of human behavior, and the cost and effort to build good models can be very high. In the real world, there often exist problem domains where the knowledge might be incomplete, imprecise or even conflicting. Often, the models are built on inflexible doctrines. This can cause the entities to behave “too perfectly” without human similarities. It has also been shown that the manual routines taught by the experts, are not necessarily the routines used by the experts themselves [2].

The use of a learning system that could automatically extract knowledge and construct a behavior model could address the problems mentioned above.

This paper presents an approach to building human behavior models automatically. The approaches employs Context-Based Reasoning (CxBR) and Genetic Programming (GP) to implement the learning artifact of a methodology called Learning by Observation, which will learn the behavior of a human merely by observation.

Learning from Observation

Inspired by how humans and other mammals learn by observation, the machine learning community has developed a number of theories on learning by observation, applied in various areas. By using learning by observation instead of traditional knowledge acquisition and development methods, the development cost and complexity could be reduced. Further, the models might be able to capture behavior patterns that would not have been found otherwise. The interest in this research is to investigate learning human behavior by observation. The intent is to use observations to learn the behavior of the observed entity. The research described in this paper defines learning by observation as follows:

> *The agent adopts the behavior of the observed entity only through the use of data collected through observation.*

To create a behavioral model with vide variety of human features there must exist an efficient modeling framework. CxBR was proposed by Gonzalez and Ahlers [4] as an efficient paradigm to model human behavior. If the model is to be created automatically, the framework needs to be equipped with a learning paradigm that will work in conjunction with the framework without disturbing the supported human features. In this research, the learning paradigm used was GP.

Context-Based Reasoning

CxBR is a modeling technique that can efficiently represent the behavior of humans in software agents. Later research showed that it is especially well suited to modeling tactical behavior (i.e. tactical decision making). CxBR is based on the idea that:

- A recognized situation calls for a set of actions and procedures that properly address the current situation.
- As a mission evolves, a transition to another set of actions and procedures may be required to address the new situation.
- Things that are likely to happen while under the current situation are limited by the current situation itself.

CxBR encapsulates into hierarchically-organized contexts the knowledge about appropriate actions and/or procedures as well as compatible new situations. Mission

Contexts define the mission to be undertaken by the agent. While it does not control the agent per se, the Mission Context defines the scope of the mission, its goals, the plan, and the constraints imposed (time, weather, rules of engagement, etc). The Major Context is the primary control element for the agent. It contains functions, rules and a list of compatible next Major Contexts. Identification of a new situation can now be simplified because only a limited number of all situations are possible under the currently active context. Sub-Contexts are abstractions of functions performed by the Major Context which may be too complex for one function, or that may be employed by other Major Contexts. This encourages re-usability. Sub-Contexts are activated by rules in the active Major Context. They will de-activate themselves upon completion of their actions.

One and only one specific Major Context is always active for each agent, making it the sole controller of the agent. When the situation changes, a transition to another Major Context may be required to properly address the emerging situation. For example, an automobile may enter an interstate highway, requiring a transition to an **Interstate-Driving** Major Context. Transitions between contexts are triggered by events in the environment – some planned others unplanned. Expert performers are able to recognize and identify the transition points quickly and effectively.

CxBR is a very intuitive, efficient and effective representation technique for human behavior. For one, CxBR was specifically designed to model tactical human behavior. As such, it provides the important hierarchical organization of contexts.

Extending CxBR with learning capabilities

Human behavior within CxBR can be categorized in two groups: action rules and sentinel rules. Rules, in this case, describe knowledge containers. These containers can contain production rules, functions, operators and complex data structures. The action rules describe the action of the agent within the specific context. Each context has its own set of sentinel rules that determine whether this context should still remain active or if it should turn over the control to another context.

To be able to implement learning by observation, it is our objective to incorporate a learning paradigm into CxBR that could learn knowledge in all the different parts of the context base. To learn specific action patterns, the learning regularly becomes somewhat of a regression problem where the model is trying to minimize the discrepancies to the human's performance. When it comes to choose the right context for the current situation, the learning process is more of a classification problem. This means that the learning algorithm needs to be able to handle both classification and regression problems. GP has successfully been used in a vast variety of machine learning problems, including regression and classification problems.

Many machine learning algorithms enforce a transformation of the search space to enable learning. For example, artificial neural networks transform the search space to a set of weights whose values are optimized during learning. Wolpert and Macready [7] conclude in the *No Free Lunch* theorem that all machine learning paradigms need to be tuned for the problem at hand to enhance their performance. In some way, the learning algorithm needs to incorporate problem-specific knowledge into the behavior of the algorithm. When the search space is transformed, prior to learning, the

knowledge to improve the learning also transforms. Instead of expert comprehension of the problem, the intellectual capacity is now focused on the learning paradigm instead of the problem at hand. A non-transforming learning paradigm supports the use of problem specific knowledge to improve learning.

The GenCL approach

The new learning algorithm presented here by merging CxBR and GP is called Genetic Context Learning (GenCL). Instead of creating the contexts by hand, we use the GP process to build the contexts. The GP's evolutionary process provides the CxBR frame with appropriate context's action rules and sentinel rules. The individuals in the genetic population are components of the context base (see figure 1). The evolutionary process will strive to minimize the discrepancies between the performances of the contexts created by GP and the human performance. The human data are the observed human performance, or rather, appropriate parts of the performance selected by the observer module.

The *Observer Module* in Figure 2 is future research to develop fully automated learning by observation. In this initial testing of the algorithm the objective is to investigate the learning module and the training data selection was done manually.

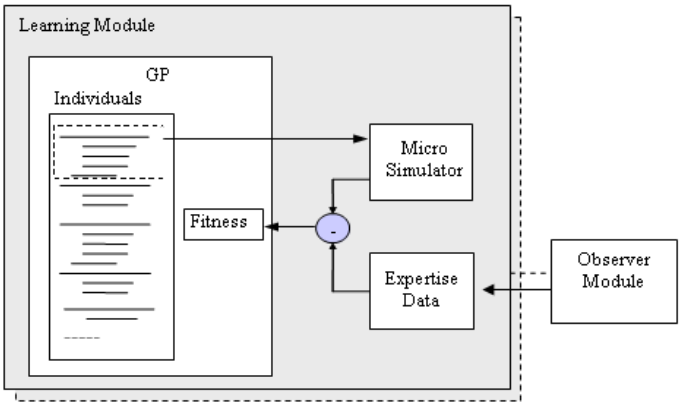

Fig. 1. Learning by Observation: GenCL

Figure 2 describes the GenCL algorithm in detail. The GP algorithm used in GenCL is a basic, tree based, source code (C-code) GP. Details of the GP configuration can be found in Fernlund [3]. The individuals are initially, randomly created. During the evaluation step, these individuals are executed in a micro-simulation, and their results are then compared with the recorded human data in the fitness function to compute a fitness value. The sequences of observed data are used by the GP's fitness function which compares the individual's behavior (i.e. execution of the source code) to the human performance. This fitness value reflects how closely

each individual replicates the observed human's actions. When the fitness value for all individuals has been established, the individuals are then ranked according to their fitness values. Individuals with a better fitness value are more likely to be selected than those with a worse fitness value. Selected individuals can be chosen to create offspring individuals through genetic operations or survive (i.e. cloned) to the next generation. The offspring and survivors from the old generation now form the new generation of individuals. This natural selection process continues over many generations, encouraging better performing individuals and pruning worse performing ones. Upon reaching a certain generation limit, or the determination that little further improvement can be expected, the process stops and the best performing individual is anointed as the winner.

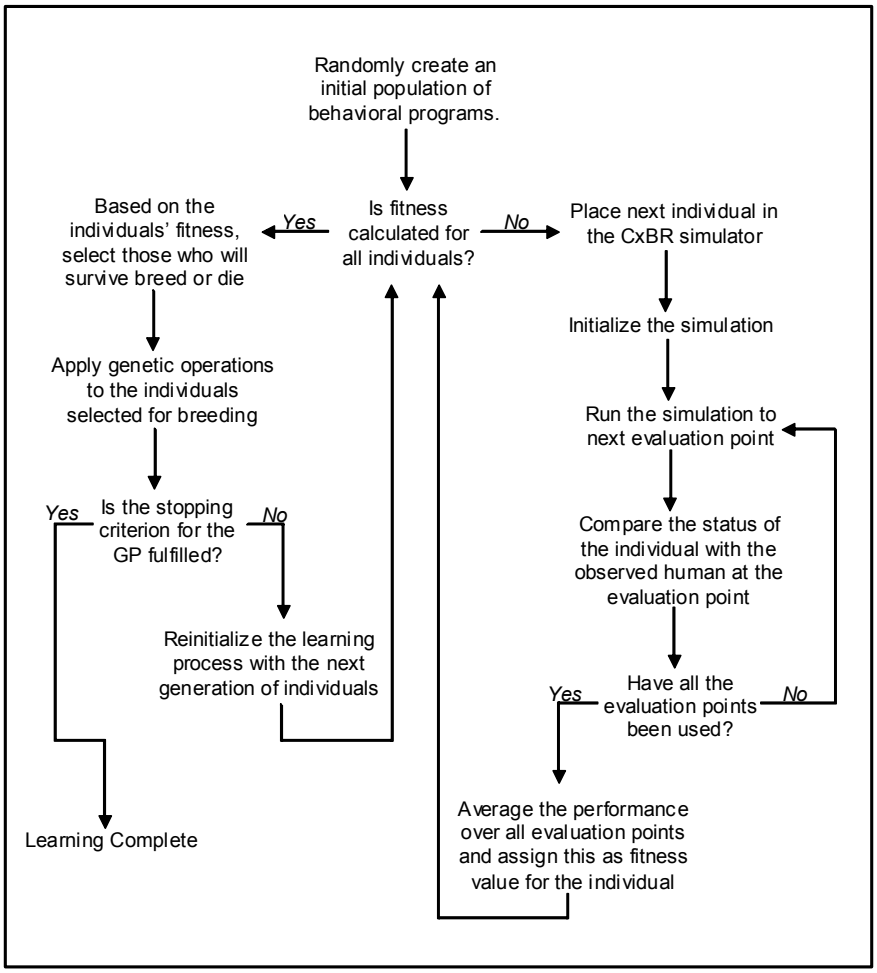

Fig. 2. The GenCL algorithm

Strategy for GP learning within CxBR

The task of implementing a learning strategy applicable at all level of the CxBR hierarchy indicates two main problems concerning the learning implementation: 1) Hierarchical complexity and 2) Interdependency among contexts and context parts.

The first problem points to the need for some sort of structure for the implementation of the learning in the different contexts. Evolving human behavior structure in all different hierarchical levels at the same time is very complex. Hsu and Gustafson [6] propose a GP learning approach where the high-level behavior would be broken down into lower level behavior that is first learned and then used when higher-level behavior is being learned. The approach is called Layered Learning GP (LLGP). Hsu and Gustafson showed promising results, where the LLGP strategy improved the learning performance.

The other problem is the issue of interdependency between the knowledge in the same level of contexts. The interdependency between different action rules in the contexts at the same level might not be that strong, since only one context could be active at the same time (i.e., mutually exclusive). Rather, the dependency is in the combined performance of the action and the context switching (i.e., sentinel rules). The collection of sentinel rules at the same context level is highly interdependent. For a sentinel rule within a context to be able to activate the context, any other contexts at the same level need to release their activation. Since the performance in one part is dependent on the performance in the other part, these parts need to be evolved together simultaneously. An approach to this problem is to use the Cooperative Co-Evolutionary strategy. In this strategy, it is possible to have different populations evolving solutions to sub-problems in parallel. When it comes to evaluation of the performance (i.e. calculation of the individual's fitness) of their joint effort, the best individuals from the other populations are included to produce and measure their performance. This research uses LLGP and co-evolution as the basic GP strategy for learning human behavior in CxBR from observation.

By using the expert knowledge to pre-define the valid contexts and sub-contexts, a coarse knowledge framework is established for the GP to evolve more complex behavior. The first use of the new approach was to model city driving. In this research, the only pre-defined structure used prior to learning was five valid contexts at three levels (**Red-Light-Driving, Green-Light-Driving, Traffic-Light-Driving, Intersection-Turning** and **Urban-Driving**). All the contexts were empty and contained no knowledge. Only the context frames were defined and their hierarchical relationship. This will provide a suitable framework for GP to operate in, but it will not restrict the behavior model in those situations. The GP learning algorithm will still be able to map number of different human behavior patterns within those situations.

Testing and Verification

The data for the experiments was collected in a full scale driving simulator where the driver sits in a car cabin and the simulated environment is projected on three walls. The objective here is not to model the best behavior or the average human behavior.

The intent is to capture the specific behavior of the current human performance (driver), no matter how good or bad his/her performance might be. The only restriction presented is the number of situations the model will be able to handle.

Five different drivers were used to collect the training sets. Consequently, five different agents were evolved with different driver behavior (i.e. each agent's performance should resemble the corresponding driver's behavior).

The data collected consists of two sets. The first set was used as a basis for the training and the second data set was used for validation. The validation set consists of new but similar situations used to evaluate the agents.

The nature of learning by observation is to let the human subject perform his task as realistically as possible and monitor and extract his true behavior. To examine the potential of learning by observation, no extra knowledge or information is added to the data. As an example, the agent will not be penalized extra during the learning process if it runs a red light. It should only compare its behavior to the human whose behavior was used to evolve it. The experimental test-bed was designed for the experts to drive a city driving route. During these 30 minute drive, the expert did not experience two situations that were identical. Human behavior is not always consistent, and the human driver might react differently to similar situations. It might be tempting to present the same situation several times and base the learning upon some average measure of the performance. The risk in these repetitive situations is that the human bases his action on prior knowledge and might not behave naturally. Conversely, letting the human act in a realistic environment with similar, but not identical, situations introduces disorder to the training data. In the worst case, contradictory data might exist in the data set. If learning by observation is to be rigorously implemented, this is an important issue. The learning conducted in this research left any disorderly data within the data set to investigate how well the CxBR and GP approach handles this.

The environment for the experiments was set up to ensure that the behavioral patterns of the drivers were neither predictable nor trivial. One feature of human behavior is unpredictability. An example of how to trigger this behavior from the drivers is found in a traffic light changing from green to yellow and then to red. If this change takes place at an appropriate distance from the car, the drivers will make a decision on whether to slow down to a stop when the light turns yellow, or continue and pass the light while yellow. The distance to trigger this diverse behavior among people seemed to be when the car is 30 meters prior to the light when driving in Swedish city traffic (speed limit of 50 km/h). When the training data sets were collected, we were able to capture this difference in behavior patterns. Even if the lights always change from green to yellow (to red) at 30 meters ahead of the light, there was a significant difference in the experts' behavior. Depending on the environment in the light's proximity and the current speed of the car, the same driver would react differently at different lights. Also, a difference among the drivers could be detected in their behavior, as described in Table 1.

When the driver does not stop, his speed is usually so high that he would pass the light when it's still yellow. One driver was very careful and stopped at all yellow lights, while others stopped at some yellow lights and ran others.

Table 1. Behavior of the five drivers at the six traffic lights that changed from green to red. S stands for stop and R for running the light while it's still yellow.

	Light 2	Light 3	Light 5	Light 6	Light 7	Light 9
Driver A	S	R	S	R	R	S
Driver B	S	S	S	R	R	S
Driver C	S	S	S	S	S	S
Driver D	S	S	S	R	R	S
Driver E	R	S	S	R	R	S

Results

A model's performance when compared with training data is by itself not sufficient to determine the success of the new approach. The key objective is to facilitate the creation of simulated agents with human behavior and to open up the possibility to capture knowledge that is hard to model and highly intuitive. The knowledge learned should not only mimic the behavior, it should also generalize the behavior and create reliable agents. Our evaluation of the GenCL approach considered following criteria:

- Generalization
- Long term Reliability
- Competitive Performance

During the evaluation, the evolved simulated car agents were inserted into a simulated environment where they operated autonomously.

Generalization

To measure generalization, the agent is inserted to operate in the simulated environment and compare its performance with the recorded driver's behavior in the same situation the driver experienced during the validation run. The validation environment refers to the new environment the drivers experienced during their second simulator run. During the drive, the agent will pass five traffic lights that change from green to red and one that changes from red to green.

Table 2. Qualitative comparison of the drivers / agents performance. S stands for stop and R for running the light while it's still yellow. OK indicates that the agent performs in accordance to the driver at the light turning green

	Light 1	Light 3	Light 4	Light 5	Light 6	Light 7
Driver A/Agent A	R/R	OK	S/S	R/R	R/R	S/R
Driver B/Agent B	R/R	OK	S/S	R/R	R/R	R/R
Driver C/Agent C	S/S	OK	S/S	S/S	S/S	S/S
Driver D/Agent D	R/R	OK	S/S	R/R	S/R	S/R
Driver E/Agent E	R/R	OK	S/S	S/R	S/R	S/R

First the qualitative behavior at the different traffic lights was examined. Table 2 shows the agents behavior at the traffic lights. Here it shows that agent A runs light 7 while driver A stops at that light. Agents B and C performs exactly as drivers B and C, respectively. Agent D, however, runs both light 6 and light 7 but driver D actually stops at those two lights. If we look at agent E and driver E, we can see that the behavior often differs at the lights. Driver E actually performs very differently at the two simulator runs. In the first run he was very reckless and ran more yellow lights than the other drivers. On the other hand, while in the validation environment, he was very careful and stopped at almost every light.

If we look at the speed and time deviations of the validation run in Table 3, the A, B and C agents perform very well. However, a slightly worse performance can be observed from agent D. Agent E, on the other hand, is not performing well at all. That agent E is not performing well in comparison to driver E is easily explained by the irregular behavior of driver E between the original training run and the subsequent simulation run.

Table 3. Speed and time deviation during the validation testing

	Speed deviation [km/h]		Time deviation [s]		Speed
	RMS	Std.Dev.	RMS	Std.Dev.	Correlation
Agent A	7.47	7.44	1.47	1.47	0.880
Agent B	7.14	6.19	2.56	1.75	0.896
Agent C	7.12	7.11	3.60	2.80	0.926
Agent D	10.5	9.23	9.10	6.78	0.712
Agent E	17.0	12.0	38.4	30.3	0.550

The results for agent A are good even when its misbehavior at one of the traffic lights is kept in the comparison. The misbehavior of agent D is more interesting to analyze since agent D's behavior is not good. An analysis of the training data shows that there was a lack of richness in D's training data. It shows that driver D actually stops at all lights abut to turn red if he is going to make a turn in the intersection where the light is located. By inspecting the code of the evolved agent D it shows that the agent found this relationship and will only stop if it will make a turn. In the validation data set driver D stopped at light 6 and 7 where he was going straight.

Long term reliability test

The long term reliability test was conducted to investigate whether the agents exhibit consistent behavior even after substantial amount of time in a simulation run. Given that GP will produce code not accessed during the training phase, and that the use of conditional statements can introduce discontinuities, the test of long term reliability is very important.

Here the five agents were allowed to operate within the simulated environment for 40 minutes, pass more then 60 traffic lights and 25 intersections. Now the agents were exposed to a variety of traffic light scenarios where none were the same as the other. To be able to compare their stability and long term reliability, their behavior was

recorded when the traffic light ahead of them was either yellow or red, since this is one of the occurrences where different behavior can be detected.

If an agent was not be stable and invokes **Intersection-Turning** when making a turn, the agent will approach the turn too fast and will actually end up beside the road. If this was the case, the agent would be stuck since no recovery algorithm is implemented for the agent to find a way back to the road. Hence, the fact that all the agents were still running after 40 minutes prove robustness in terms of **Intersection-Turning**.

Since the traffic lights now change their states at different distances (i.e. the lights are time scheduled and not related agents distance) and agents might approach the lights at different timings and speeds, it is difficult to make an extensive statistical analysis of their behavior. However, Table 4 shows a simple compilation of the agents' behavior when they approach lights that is either yellow or red. Two different events occur: the light turns from green to red or from red to green.

Table 4. Agents' Long term behavior

	Light turning Red			Light turning Green
	Stopping	Avg.Dist	Std.Dev	Correct behavior
Agent A	20/20	34.7	12.9	20/20
Agent B	22/22	8.04	1.95	22/22
Agent C	25/25	5.89	1.03	8/8
Agent D	31/34	4.50	1.31	6/6
Agent E	22/22	13.5	0.551	11/11

A qualitative measure could be performed of the agents' action when the lights turn red. The stopping column in Table 4 shows how many lights the agents stop at, compared to the total number of lights passed turning red. All the agents, except agent D, stop at all lights turning red. Agent D runs three lights when they turn red late. Investigating the results more thoroughly, it shows that if the lights turn red when the agent is further away than 27 meters the agent will stop and the occasions where the lights turn red when the agent is closer than 23 meters the agent will run it. Even if the agent some times runs the light, it is consistent and acts the same in similar situations.

As the agents come to a stop at the red lights, a comparison could be made where they actually stop in font of the lights (Avg.Dist. in Table 4). Table 4 shows that all the agents except agent A stop at almost the same distance every time and, therefore, their standard deviation on the stopping distance is small. Agent A stops at different distances almost every time. The surprising fact is actually that the other four agents manage to generalize so well that they stop at approximately the same distance, even if the time of light change is different. All the training data presented to the agents during learning, lights changed their state when the driver was 30 meter prior to the light. Hence, all the agents stopped consistently at the same distance during training (approximately thirty meters after the light turns from green to yellow). So, the most obvious thing is not for the agents to generalize as well as they have, but rather to stop in the vicinity of thirty meters after the light change from green to yellow.

The final observations on the agents' long-term performances are their behavior when approaching red traffic lights turning green. Two observations can be made

here. The first thing that reflects correct behavior on the part of the agents is that they do not stop at the red light when they are too far from the light. The other behavior to investigate is that they lower the speed as they get closer and that they pick up speed when the light turn green. The column that describes the correct behavior at a light turning green in Table 4 compares the number of correct behaviors to the total numbers of lights turning green exposed to each agent. All the agents show a correct behavior all the time as they approach a red light about to turn green.

This test has shown that the agents show consistent and stable performance throughout the long term stability test. Four of the five agents even generalize their traffic light driving better than could be expected in light of the training data presented during the learning phase.

Test of Competitive Performance

In order to determine how useful the automatic creation of simulated agents through GenCL is, two agents were developed by an independent source in the traditional way [1]. Here a knowledge engineer interviewed and rode an automobile with two drivers. The two drivers were drivers C and D that earlier had been driving the simulator runs resulting in C's and D's training and validation data sets.

The knowledge engineer knew that his model would be compared to those developed by the GenCL system and was told to focus on the behavior patterns so far implemented by the GenCL. Hence, the prerequisites for the knowledge engineer were the same as for GenCL (i.e. the same empty context structure). The task was for the knowledge engineer to collect knowledge through interviews and by observing the humans drive a real car. After the knowledge was collected and analyzed, two agents were developed and implemented to run in the same CxBR framework as did the agents developed by GenCL. Note that the knowledge engineering was done by an independent researcher without any influence from the ones developing GenCL. Now, the two different approaches to build human behavior models, implemented in simulated agents, could be compared. The agents developed by the knowledge engineer were exposed to the same scenarios as those created through GenCL and their behavior could be compared to the driver's behavior. Table 5 compares the agents developed by the knowledge engineer (KE) and the GenCL agents to the driver's behavior in the Driving Simulator.

Table 5. Comparing GenCL and Knowledge Engineer agents in the validation environment

	Speed [km/h]		Time [s]		Speed
	RMS	Std.Dev.	RMS	Std.Dev.	Correlation
KE agent C vs. Driver C	8.52	8.38	4.05	3.10	0.902
GenCL agent C vs. Driver C	7.12	7.11	3.60	2.80	0.926
KE agent D vs. Driver D	9.02	8.64	7.43	7.21	0.876
GenCL agent D vs. Driver D	10.5	9.23	9.10	6.78	0.712

Comparing GenCL agent C and KE agent C with driver C, the agent evolved by GenCL performs slightly better than KE agent C. Comparing D agents to driver D,

the GenCL agent performs slightly worse than KE agent D in the validation environment. The interesting result here is that the GenCL agent is able to perform almost as well as an agent developed by traditional means, even when the GenCL agent D was affected by the lack of richness in the training data.

This test show that the learning and generalization capabilities of GenCL are able to create an agent performing, at least as well as an agent developed through traditional means.

Conclusions

The results presented in this paper show that the new GenCL learning algorithm is able to evolve human contextual knowledge. The agents evolved further show the ability to generalize the behavior, stable performance and performance competitive with agents developed manually by human developers.

The results presented here have also shown the ability of GP to produce knowledge in all the different parts of the CxBR's context base. GP has been able to evolve knowledge in the action rules of the contexts and also the knowledge in the sentinel rules. Since GP produce knowledge that is interpretable and understood by humans knowledge could be extended and refined either prior to, or after learning is conducted. This feature of GP also enables investigation of the knowledge learned. Hence, the models evolved could be investigated to discover hidden or unknown behavior patterns of the human observed.

Further development of this algorithm to fully implement learning by observation could probably ease the development of human behavior models in simulated agents.

References

1. Bergström J., (2003) "Context Based Reasoning: knowledge engineer vs. machine learning in the knowledge elicitation phase", Degree Project, Dalarna University, Sweden.
2. Deutsch, S. (1993) "Notes Taken on the Quest for Modeling Skilled Human Behavior" Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, March 17-19, pp.359-365
3. Fernlund, H. (2004), "Evolving Models from Observed Human Performance" Doctoral dissertation, Department of Electrical and Computer Engineering, University of Central Florida, Spring, 2004.
4. Gonzalez A. J. and Ahlers, R. H. (1998) "Context-Based Representation of Intelligent Behavior in Training Simulations" Transactions of the Society for Computer Simulation International, volume 15, no 4, December 1998
5. Gonzalez A. J. and Ahlers, R. H. (1998) "Context-Based Representation of Intelligent Behavior in Training Simulations" Transactions of the Society for Computer Simulation International, volume 15, no 4, December 1998
6. Hsu, W.H., Gustafson, S.M. "Genetic Programming for Leyered Learning of Multi-agent Taks", Proceedings of the Genetic Evolutionary Computation Conference, GECCO 2001, San Francisco, CA, July 9-11, 2001
7. Wolpert D.H. and Macready, W.G. (1995) "No free lunch theorems for search", Technical Report SFI-TR-95-02-010, 1995.