Agents for Education

Bringing Adaptive Behavior to an Internet Learning Community

Thesis Proposal

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Department of Computer Science

Jordan B. Pollack, Advisor

by

Elizabeth Ida Sklar

April 1999

Abstract

Can *adaptive behavior* be useful in a human learning environment? The work presented here proposes that introducing adaptive agents to a human learning community can be effective in several ways:

1. to guide selection of appropriate problems for human learners,
2. to embody "self-organizing" curricula,
3. to provide simple student models,
4. to impersonate human learning partners, and
5. to facilitate indirect human-human interaction.

Taking a population-based approach and using evolutionary computation methodologies, we bring two types of software agents into an Internet learning community. Inside our CEL (Community of Evolving Learners) system, humans play competitive, educational games, openly or *incognito*, with each other and with software agents, while the populations of learners – human and agent – advance.

The overarching goal of this research is to investigate the various factors that contribute to this type of environment and their effects on both human and machine learners. We wish to demonstrate that the progress of each student can be tracked and that the agent populations adjust according to changes in human behavior, taken both on an individual and aggregate basis. Hopefully, the result will be that humans participants frequent the system over time and become active members of the learning community.

1 Introduction.

With the advent of the Internet, humans have found new and exciting ways to communicate. The explosive popularity of electronic mail, news groups, instant messaging and on-line chat is proof that, like the telephone, computer networked communication is here to stay.

These new forms of human interaction allow us to establish the notion of a *virtual community*, wherein humans in diverse places and time zones can exchange thoughts and share experiences just as in a traditional community, but the need for co-located participants is now eliminated.

Humans are not alone in populating the Internet. *Software agents* also inhabit cyberspace — small computer programs that are designed to help human users accomplish specific tasks [53, 54]. Agents have been created to perform many different functions; and Internet agents are gaining prominence as browsing assistants [52], matchmakers [27, 46], recommenders [5] and filterers of email [31, 49] and news group messages [47]. Some agents use *machine learning* techniques to adapt their behavior to the needs of individual users [71, 5].

The work presented here brings adaptive software agents into an Internet environment where agents help human users learn and, in turn, learn how to do their own job better through their interactions with the humans. Inside the Community of Evolving Learners (CEL), humans play competitive, educational games, openly or *incognito*, with each other and with software agents; and the learners – human and agent – advance.

Within the CEL framework, there are two types software agents: (1) *bitbots* and (2) "secret" agents. Bitbots represent little bits of knowledge

to be learned; a *population* of bitbots is a problem set — i.e. the content of a game. For each human user, an individualized population of bitbots is created and adapts based on that human's changing needs.

Inside CEL, pairs of humans engage in competitive games. Their individual bitbot populations are merged, and the composite sets of bitbots act as proxies in their matches.

The CEL system attempts to create appropriate matches between humans in such a way as to stimulate learning. However, CEL is an open system on the Internet and does not control who is logged in. It is possible that for a given human at a given time, no appropriate human partner is also connected. Therefore, CEL provides "secret" agents that act as substitute learning partners and emulate the behavior of human players. These agents also adapt, adjusting to the needs of the aggregate human population.

To enable learning in both types of agents, a population-based approach is taken and evolutionary computation techniques are applied. Some of the methodologies were used in earlier work [30, 29, 80, 81], where the notion of *co-evolution* between populations of humans and software agents was introduced.

In *co-evolutionary machine learning* [36, 3, 58, 73, 79, 41, 69], two (or more) populations of agents compete against each other. In successful applications of this technique, an "arms race" spiral develops where each population spurs the other to advance, and the result is continuous learning for both populations. Work examining the dynamics of co-evolutionary learning environments has given insight into phenomena that contribute to success [15, 10, 69, 24]; in particular (1) it is important to select appropriate experiences for individual learners — pairing a novice to compete with an expert

will not help either player advance; and (2) it is crucial to prevent collusion between members of competing populations — two players agree to "draw" each other so that neither will be eliminated, but also neither will learn.

The work presented here carries these insights into the realm of human learning. The co-evolutionary approach is applied in three ways. First, a population of bitbots co-evolves with each human user. Second, a population of secret agents adapts with the population of human users taken as a whole. Third, humans compete against each other in the games they play. It is recognized that the use of competition in education is controversial. However, if participants are anonymous and success rates are controlled, many of the more contentious issues may be superseded in favor of the more powerful motivational aspects that competition can provide.

2 Contribution.

The main contribution of this thesis will be to investigate the effects of adaptive environments on both human and machine learners, demonstrated in an Internet community where humans play competitive, educational games with each other and with software agents, and where the learners – human and agent – are both evolving. We hope to show that the progress of a student can be tracked, that the system can adapt in its selection of appropriate competitors and game content, and that the proper choices result in human participants becoming active members of the learning community.

There are several aspects of CEL that serve to support these claims and distinguish this project from other Internet communities and intelligent tutoring work:

1. *Adaptive behavior approach to problem selection.* A new type of agent is introduced, called a *bitbot*, that represents little "bits" of knowledge to be learned — and is actually the content of each game. A population of bitbots constitutes a problem set. For each human user, a population of bitbots is instantiated, and, using evolutionary programming techniques, adapts as the user progresses (see section 6.1). *We believe that we are the first to apply adaptive behavior methodologies to an intelligent tutoring system.*

2. *Self-organizing curriculum.* While the performance of each human drives the selection of individual bitbot populations customized for each user, for the aggregate human population, the sequence in which bitbots are chosen may generalize into an ordered, hierarchical list. Each tier could be considered a step within an incremental, or staged, learning environment, where the stages are evolved, not engineered. *We believe this to be a new type of "self-organizing curriculum", one that emerges as the system is used — feasible because of the mass human data collection opportunities enabled by the Internet.*

3. *Simple, evolving student model.* A student's performance with the bitbots serves as a model of that student's abilities. This model evolves with the student and requires no domain-dependent engineering to build (aside from defining the bitbots themselves) (see section 6.2). This approach is quite different from more traditional user modeling techniques, which are typically highly engineered to the task domain (see section 4). *We believe that this simple, evolving student model will be sufficient to guide appropriate problem selection and opponent choice.*

4. *Playgroup control.* Humans are placed in appropriate *playgroups*, guided by their student models and a prediction of how they will perform against each other (see section 6.3). They can only challenge *playmates* (opponents) from this group, whose membership is controlled by our server with the aim of producing a desired outcome (i.e. *win rate*) for each member across a series of interactions. This notion of only giving players access to appropriate playmates distinguishes CEL from other game playing sites. Typically, users can see everyone else who is logged in at a given time (e.g. `games.yahoo.com`). *We believe that exercising control over a user's playgroup will allow us to experiment with various membership compositions, to help determine what "appropriate opponent choice" really means.*

5. *Indirect player interaction.* CEL users communicate indirectly, with agents acting as mediators. When two users play a game in CEL, each communicates directly with his/her bitbots; and each user's bitbots pass "moves" to their opponent's bitbots. In other Internet communities (e.g. MUD's), users communicate directly in a "chat" environment; this is a controversial setup for children, due to privacy and safety concerns for young Internet users, as well as curricular issues like ensuring that participants stay "on task". *We believe that CEL's method of indirect, mediated human-agent-human interaction addresses these concerns effectively.*

6. *Creative, graphical, individualized player identification.* CEL is a graphical environment, and players are identified to others only through 2-dimensional icons, called *IDsigns*. A simple web-based tool allows users to create and edit their own IDsigns. In most Internet communities,

users are known to others by name (usually an alias). *We believe that allowing users to create and edit their own icons graphically helps spark interest for young players, provides an outlet for their creativity, as well as preserving user anonymity.*

7. *Secret agents as learning partners.* When no appropriate human opponents are available in the system, "secret" software agents are instantiated to act as substitutes (see section 6.4). It may be important for these agents to behave intelligently enough for a human user to believe she is interacting with another human. Neural networks control these agents, trained using a novel approach that takes advantage of human behavior observed in the system. *While we do not claim to be able to pass a Turing Test, we believe that providing "secret" software agents to emulate human learning partners enables the system to maintain the desired playgroup control (as above) without making users wait for appropriate mates to log on to the site.*

Finally, a disclaimer: there will be no attempt to prove that *as a direct result of using the system*, learning has occurred on the part of the humans who participate. The skills tested in this initial implementation of CEL are fundamentals (like spelling and arithmetic); the target user group is primary aged school children, and participants will be exposed to many other stimuli for advancing these skills outside of CEL.

It is hoped that future work will extend the games in CEL beyond fact-learning applications. But, just as a child must be able to spell before she can write, or be able to add before she can manipulate differential equations, CEL begins with simple games. A longterm goal is to use the CEL framework to

enable more complex problem solving scenarios and to host further human-human and human-agent learning experiments.

3 Motivation.

In the last two decades, the invention and rise in popularity of personal computers has brought about a new and burgeoning market for educational software. Subsequently, with the introduction of computers into schools, the market has continued to expand and educational software has infiltrated classrooms, with mixed results. Today, schools are being "wired" at a rapid rate, giving teachers and students direct access to the Internet. Yet, with all this vast and varied hardware and software, the predominant paradigm involves one student sitting at one computer using one program — *alone*.

However, it has been shown throughout history that group learning can be very effective [84, 83, 40], especially when participants *cooperate*. In contrast, the role of *competition* in human learning, particularly in formal education, is controversial [63, 43, 40, 39]. Yet competition is a central organizing principle in western society. Athletic events, political campaigns, legal cases, industrial battles, military encounters, Nobel prizes — all are "races" to be won or lost. By eliminating competition from schools, are we ill-preparing our children to thrive as adults in the "real world"?

The work presented here explores the use of anonymous competition, customized to the individual, in a human learning environment. Perhaps if learners' identities and ages are kept hidden and win rates are controlled, then many of the more contentious issues associated with the use of competition in education may be superseded in favor of the more powerful motivational

aspects that it can provide.

3.1 Machine learning.

The general idea of *machine learning* – where computer programs advance, developing better and more efficient ways to accomplish given tasks – has been around since (at least) the 1950's and has often been thought of in the context of game playing [75, 57, 26, 9, 90, 22]. Games are a good domain for such research, since they typically have a fixed goal and a finite set of predefined rules that allow a player to achieve the goal within a reasonable amount of time. Chess, checkers and backgammon have been a few of the more popular games used for these studies.

A game can be played by using a certain strategy, or set of strategies — a method and order for applying the rules of the game, with the intent of achieving the fixed goal. If we sat down and enumerated all the possible ways of playing a game, the result would typically be a huge list. So the question becomes a matter of *search*. Given a very large list of possible strategies, how can we find the ones that will result in achieving the game's goal?

The following is an *evolutionary programming* approach [26, 37, 45]: rather than try to engineer a winning strategy, enumerate a manageable number of strategies, use these to play games and see how well they do. Then keep the strategies that do well and use *selection* and *reproduction* techniques to replace the ones that do poorly with other strategies that have not been tried yet. Using this method, a population of successful strategies is built up gradually. At any time, the population will represent some ways of playing the game; eventually, the population will contain the optimal way(s).

3.2 Human learning.

A key concept in human learning is "learning by doing." *Constructionism* [64, 65, 72] is a formal definition of this notion applied to education. For humans, motivation is of primary concern. While "practise makes perfect," students need to be motivated to practise in the first place — and to practise correctly, not to practise mistakes (otherwise the mistakes will be learned instead).

How can students be motivated to practise? Today's computer games are filled with fancy animations and nifty audio. In order to vie for a child's attention, a software application has to be able to compete with this standard — or else proffer something completely different. Humans are naturally social creatures. If a software application offers a learning environment that is a *community*, then humans should be encouraged to participate purely by their innate desire for human contact, even virtual human contact. The amazing popularity of text-based MUD's[1] is such an example, and many have used this to their advantage in building successful educational MUD's [12, 33, 23].

If an environment encourages students to keep practising, then learning should follow naturally. In an artificial co-evolutionary learning situation, where two (or more) populations of agents are advancing, the learning environment consists of other learners — students are *each other's* teachers. We posit that this is also the case in our CEL environment, and hypothesize that some of the features that contribute to the success of a co-evolutionary learning environment will also be pertinent here. As such, our environment should be constructed to (1) provide appropriate experiences for individual

[1] Multi-User Dungeon/Dimension/Domain

learners; and (2) prevent collusion between participants.

What does "appropriate experience" mean? Fundamentally, if a student is given problems that are too hard, then he will answer them by guessing and will not learn, or may even learn the wrong answers; if a student is given problems that are too easy, then he will get bored when answering them and may provide inaccurate answers due to lack of attention; if a student is given problems that are "just right," then he will be challenged appropriately and learning will take place. In an environment where students are learning from each other, the notion of appropriate experience also applies to matching students with proper learning partners. This concept has been formalized for artificial learners in the Meta-Game of Learning (MGL) hypothesis [69, 80] and for human learners, is related to Vygotsky's "zone of proximal development" [95] and to "scaffolding" [38] in intelligent tutoring systems.

How can "collusion" be prevented? One problem with a MUD is that activity and communication are not controlled. Users are free to "say" anything they want. In some educational contexts – such as learning creative writing – open communication is desirable and necessary. But in others – such as learning multiplication tables – this is not necessary and may not even be desirable, because bored students will be inclined to stray from the curricular task and shift the discussion to other topics[2].

Another issue with unrestricted communication, especially on the Internet, is that parents are worried about who their children are communicating with. Though everyone in a community may be using an alias, an anonymous exchange of inappropriate ideas or language may still occur. Providing a safe

[2]This could also occur in the creative writing domain.

environment in which young students can interact is of primary concern.

If contact between participants is mediated artificially, where software agents enable indirect human-human interaction via direct human-agent interaction, then the agents can force human participants to stay "on task", thereby preventing collusion and addressing safety issues as well.

4 Some related work.

This work finds itself at the intersection of several areas of human and machine learning, particularly intelligent tutoring systems (ITS), student modeling, collaborative learning and adaptive behavior. These fields encompass a vast literature which will be reviewed thoroughly in the thesis. Many pertinent references are cited throughout the text. Some others are cited below.

Seminal work in the ITS field began with frame-based tutoring systems [11] and continued to focus in areas like constructing rules [2] and modeling student's misconceptions [93, 85]. Some of these ideas have been developed into systems tested in laboratories, classrooms and work places [42, 77].

More recent work has continued to explore these areas in more depth. Student modeling has been aided by statistical techniques [56, 6, 94] as well as artificial intelligence methods like case-based reasoning [1]. Other work has examined customizing to the needs of individual students in web-based systems, for example ELM-ART [13], CALAT [62] and MANIC [86]. These require considerable domain engineering on the part of the system designer/programmer and significant psychological testing to reveal the granularity of a single curriculum area.

Others are working to enable Internet learning communities; projects such

as CoVis [66], KIE [8], Virtual Classroom [92], Belevedere [87], PuebloMOO [96] and MOOSE crossing [12] are a few of the more successful examples. However, we believe that no one else is examining the role of competition in this type of virtual learning environment.

5 A brief tour of CEL.

CEL is an Internet-based system where students engage in two-player educational games. These games are easy to use and provide practise at basic skills (e.g. keyboarding, arithmetic, geography, spelling). The system is built using HTML, CGI-bin, C and Java and is accessible from a web browser like Netscape[3].

Users log into the system with a personal user name and password. Once inside, users are represented by two-dimensional icons called *IDsigns* which identify them to other users. In order to maintain the anonymity that some CEL experiments require, and for reasons of privacy, the user name (and password) are never shown to others. Users access a tool called the *IDsigner* to create and modify their IDsigns (figure 1a). Once a user is connected to the system, she selects the game that she would like to play (figure 1b).

[3]currently the only browser fully tested

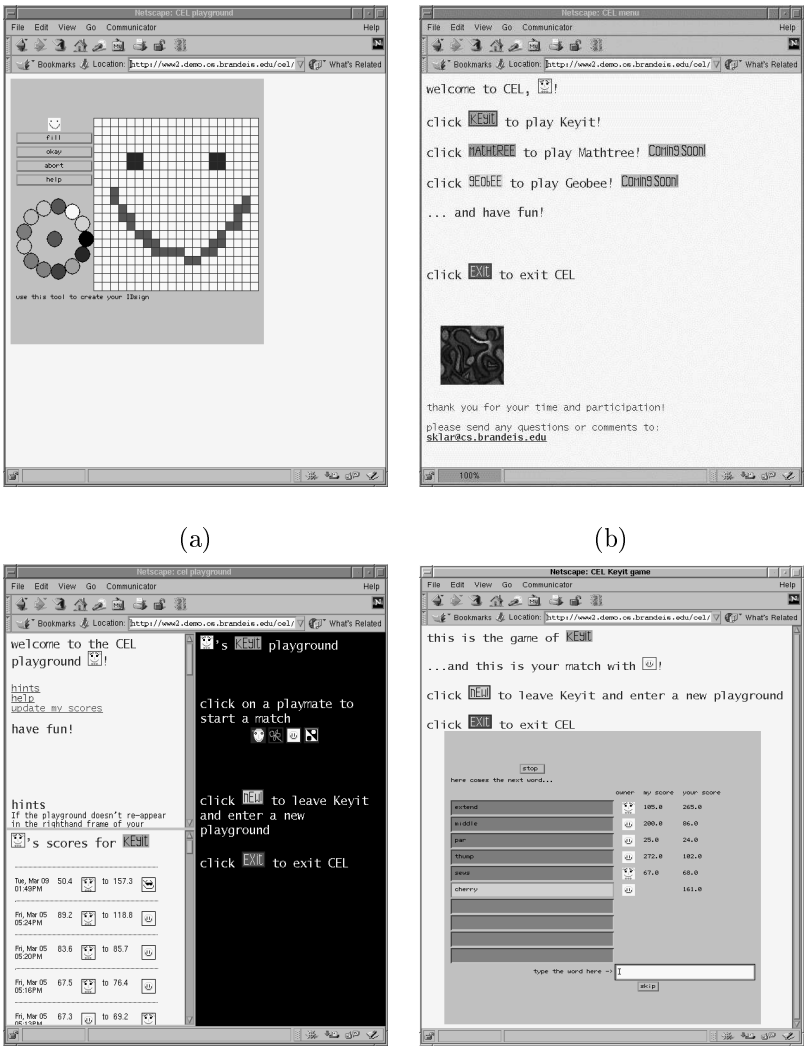


Figure 1: The CEL system.

Then the user is placed in an open *playground*, where she is shown a matrix containing IDsigns of other users who are currently logged into the system and are playing her chosen game (figure 1c). These are her *playmates*; together they comprise her *playgroup*. By clicking on a playmate's IDsign, the user challenges the playmate to a match.

During the course of a game, feedback is provided to each user, updating the moves of the opponent in near real-time. However, the match may not occur simultaneously at all if, for example, a network link is slow or one user is interrupted. In this case, the system provides to each user whatever moves are available from the opponent.

An example of the keyboarding game called *Keyit* is shown in figure 1d. In this game, users are each given up to 10 words to type as fast as they can, maintaining 100% accuracy. The two players need not type the words at the same time. The timer begins when a player enters the first letter of a word, time is measured using the system clock on the client's computer and score is reported in hundredths of a second.

Once a user has completed her game, her browser returns her to the playground and she is free to engage in another match with another (or the same) playmate.

6 Approach.

CEL takes an *adaptive behavior* approach that sets it aside from other intelligent tutoring systems. The issue of providing students with appropriate learning experiences becomes a question of finding the right set of knowledge bits (i.e. bitbots) for that student to interact with. This question can be re-

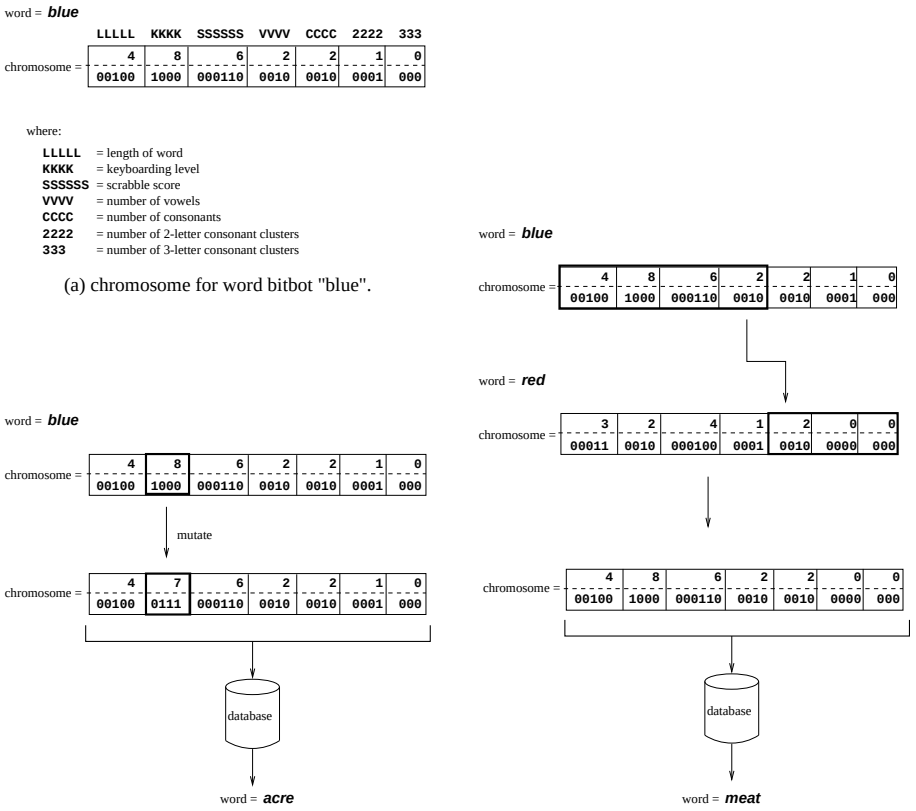
duced to a search problem – similar to finding the right game playing strategy – and is attacked with a co-evolutionary method. The game is to challenge the student, and the goal is to get the student to provide the right answer a certain percentage of the time. A population of bitbots is generated and evaluated according to how well they achieve the goal — of having the student provide the right answers a certain percentage of the time. When a student has interacted with a small number of bitbots, the population is replenished for further interactions, keeping some bitbots and replacing others.

Note that the students are learning. A set of bitbots that achieves the desired goal at one point in time may not solicit the same result several weeks or months later. This means that one fixed solution will never be found because the search is trying to optimize to a moving target. Hence *co-evolutionary learning*, wherein the two populations – students and bitbots – are learning simultaneously and through their interactions with each other.

6.1 Problem selection.

Based on earlier work [80], a simple agent called a *bitbot* has been developed, using concepts adapted from genetic algorithms and evolutionary programming [26, 37, 45]. A bitbot represents a small "bit" of knowledge that a student must learn, encoded in a *genetic* bit string. Bitbots currently come in two varieties, to support word and number games.

The definition of a bitbot is domain-specific. An example word bitbot is shown in figure 2a, for the word **blue**. The word bitbots are used for both spelling and keyboarding games, so there is an attempt to capture features of a word that might make it easier or harder to spell or type and to use these to define seven *genes*, which are concatenated to form a *chromosome*.



(a) chromosome for word bitbot "blue".

(b) sample mutation of "blue".

(c) sample crossover of "blue" with "red".

Figure 2: A sample word bitbot.

The selection of these features is based merely on empirical evidence, with the exception of the "keyboarding level."[4]

A population of bitbots constitutes a problem set. Evolutionary programming techniques are used to define an initial population of bitbots for interacting with a new user and to produce subsequent populations of bitbots as the user progresses.

Various techniques will be tested for forming this initial population and for

[4]There is a standard order in which keys are introduced to students learning typing. The keys are presented in eleven groups; each group is considered a "keyboarding level" here.

producing subsequent generations. In general, an initial population is chosen at random; however, it may be more effective to impose some limitations on this initial selection. If the first set of bitbots seen by a new user happens to contain members that are all too hard or all too easy, then the user may not get a good first impression of the system and may be uninterested in participating further. Rather it may be more effective to form an initial set based on a uniform distribution amongst the various values of bitbots.

Reproductive operators like mutation and recombination have all proven effective for various tasks using genetic algorithms. Currently, mutation and single-point crossover are being used to produce members of each next generation of bitbots. An example is illustrated in figures 2b and 2c. The numeric chromosomes from the current population are transformed into chromosomes for the next generation. For each new (valid) chromosome, a corresponding word is located in a database. The entire set is the content of the user's next game.

The selection of appropriate bitbots for each user to interact with is crucial to the success of the system — bitbots that are too hard will frustrate the user; bitbots that are too easy will bore the user. Since the CEL system exists on the web, where thousands of other sites are just a click away, these issues are perhaps even more important than with other applications. The system has to grab a student right away, challenge her enough to keep her interest and entice her to return.

A human's performance drives the selection of bitbots. For the aggregate human population, the sequence in which bitbots are chosen may generalize into an ordered, hierarchical list. Each tier could be considered a step within an incremental, or staged, learning environment [48], where the stages are

evolved, not engineered. This can be considered a *self-organizing curriculum*, and this technique could have significant impact for future intelligent tutoring systems.

6.2 Student modeling.

For each student, her performance with a set of bitbots gives an indication of how well that student has acquired those bits of knowledge that the bitbots represent. This is viewed as a simple student model.

The model is a by-product of the design of the CEL system, and it is obtained through a student's normal interactions with the system. No domain-dependent engineering is involved in constructing this model, outside of that which is done to define the bitbots themselves. This type of model is different from others found in the literature, e.g. [56, 6, 94, 1]. Experiments will be conducted to see if such a simple model is useful — a model that is based exclusively on a user's performance with the system, what the user did rather than how the user did it.

The most common function for a user model is to guide subsequent interactions between the particular user and the system. In CEL, this has ramifications in two areas. First, it can effect which bitbots are selected for the user to interact with. Second, it can guide matching of appropriate opponents. Each of these is addressed separately in sections 6.1 and 6.3, respectively.

Preliminary experiments are being conducted based on the data from an earlier experiment [30, 29], which produced a large database of performance statistics between human Internet users and software agents playing a simple video game called Tron. The agents were artificial players, controlled by

genetic programs. The results of over 200,000 games comprise this data set, played by approximately 4000 human users and 3000 agents.

In this context, a user model is conceived for each human player as a large vector containing the averaged results over all encounters with every agent. This is a sparse vector because, while there are over 3000 agents existing, most humans played an average of about 87 games (although 19 users have played over 1000 games apiece), which means that most humans met less than 3% of the agents.

A CEL environment has been built for Tron – called the Tronament – that has the same system architecture as the educational CEL, but participants engage in games of Tron, rather than educational applications like Keyit. This setup is being used to experiment with the vectors described above, to see if this type of model can be used to affect three areas: problem selection, playgroup formation, secret agent usage. This will be a useful first step, since there already exists a substantial data set to guide the processes as well as a large user base playing Tron. The assumption is that results obtained from these experiments will carry over directly to the educational applications.

6.3 Player clustering.

Players are clustered into playgroups containing potential playmates. It is critical to form a playgroup for each player such that the playmates therein can provide appropriate challenges. Of course, given the unpredictability of human behavior, it will be impossible to surmise precisely what a potential playmate's behavior will be, but an educated guess is made. A record of predictions and outcomes will be maintained in order to obtain a measure of the reliability of the predictive mechanism, as it applies to each user.

Taking this prediction reliability rating and each user's student model as input, users are clustered and playgroups are formed. These groupings are highly dynamic, as users enter and exit the system and as games are played and the student model and prediction ratings change.

A playgroup is represented conceptually as an undirected graph, where each player is a vertex in the graph. Edges are drawn between players who are considered to be appropriate *playmates*. Edges are updated as players enter and exit the game and as games are played and users progress.

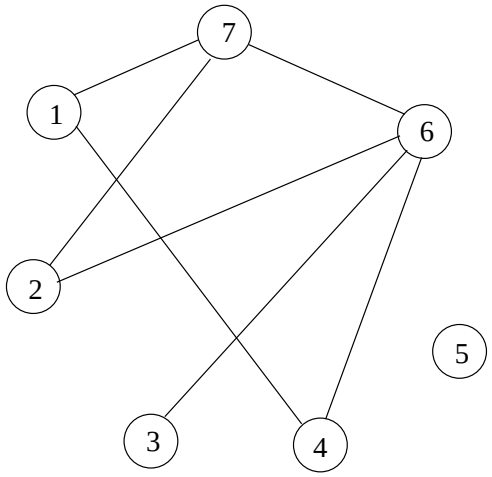


Figure 3: The graph of a playgroup.

Player 1's playgroup contains players 4 and 7; player 2's playgroup contains players 6 and 7; player 3's playgroup contains player 6; player 4's playgroup contains player 1 and 6; player 5 has no playmates; player 6's playgroup contains players 2, 3, 4 and 7; and player 7's playgroup contains players 1, 2 and 6.

An example is shown in figure 3. A player only sees those players that are in his playgroup. In the example shown, this means that even though players 1 and 2 are both connected to the same game at the same time, they do not see each other's IDsigns in their playgrounds because the game server does not deem them to be appropriate playmates. Connections are bi-directional,

so (e.g.) if player 1 sees player 7, then player 7 also sees player 1. However, links are not transitive: players 1 and 7 see each other, players 7 and 2 see each other, but players 1 and 2 do not.

This notion of only letting players see appropriate playmates distinguishes CEL from other game playing web sites. Typically, users can see everyone else who is currently logged in; this is the method used at the popular site `games.yahoo.com`, for example. But in order to maintain control over success rates in CEL, users are only given the opportunity to challenge others who are near to their own skill level.

Two algorithms are being explored for defining playgroups, (which is equivalent to determining the edges of the graph in figure 3). Each uses a different general approach, one *absolute* and one *relative*.

In the first approach, all players are ranked according to an absolute scale and only vertices of players whose ranks are within a certain epsilon (ϵ) of each other are connected. This algorithm maintains an auxiliary index on the list of players, sorted by rank. Edges are drawn on the graph by sliding a window of width $2 * \epsilon$ down the indexed list of players and connecting all players that are inside the window. The cost of this method is $O(r * n^2)$, where r is the time it takes to calculate the rank of each player and n is the number of players in the graph.

The second method computes a relative distance between every pair of players in the graph and only connects vertices of players whose distances are within a certain epsilon (ϵ) of each other. The cost of this method is $O((dn)^2)$, where d is the time it takes to compute the distance between any two players and n is the number of players in the graph.

The first method will run faster (unless $r \gtrsim d^2$, which is unlikely). How-

ever, there are two reasons to pursue the use of the second method. First, relative matching may produce more accurate outcomes in terms of shared experiences that are beneficial to both players. Second, in some domains, it may be difficult to define an absolute ranking. As such, effort will be expended to devise an efficient algorithm for carrying out the second method.

In addition to these methodologies, known clustering algorithms will be applied: Cobweb [25], Unimem [50], ID3 [70], LSI [18] and other statistical methods. A comparison of the performance of these algorithms, measured in terms of run-time and effectiveness of output, will determine the method that works best for CEL.

6.4 Secret agents.

When no appropriate human playmates are available in the system, "secret" software agents are used as substitutes. It may be important that these agents behave intelligently, to pass a minimal Turing test so that a human user will believe she is interacting with another human. Although for some users, it may not matter if their opponents are humans or agents; experimental data may shed light on this issue.

Since all communication in CEL is indirect, mediated by bitbots, there are no natural language issues to consider. This is often the most difficult obstacle to overcome in a Turing test. However, there are other critical issues involved, for example to build an agent that will not win all the time, but will be consistent, that will evolve and appear to be learning itself, but at an honest pace, and that will maintain a believable presence in the system. Since the actions of humans interfacing with the system are being observed, it is possible to create such an agent by emulating these behaviors.

At the playground level, there are two goals: (1) to produce a specific win rate for the agent, adapting over time, and (2) to maintain an active and human-like presence in the system. This means the agent should regularly log in and out of CEL during the same time of day, as if in a certain time zone; and the agent should stay logged into CEL for a variable amount of time. It also means that the agent should not only wait to be challenged by its playmates, but also challenge others.

At the game level, the agent's behavior will probably be more closely watched by the humans, since the agent's moves directly affect the performance of its human opponent. Two methods for controlling game behavior will be tested. The first way is simply to specify a win rate for the game, and let the human user drive. After each move by the human, the agent produces a move in reponse, maintaining the specified win rate.

A second method of doing this is to use a neural network to control the agent. The network is trained using supervised learning techniques. This idea is being explored using the Tron data set (described above in section 6.2). In addition to storing the outcome of each game played, the actual moves of most of the games have also been saved. This allows replaying of any of these games. A neural network that plays Tron has been created and trained by replaying games of individual players.

7 Experiments.

Experiments and data analysis will focus in two main areas: human learning and machine learning. The following data will be collected:

- results of games played — for each game: *timestamp*, *player id* (human), *opponent id* (human or agent), *bitbot id's*, *score* (for each bitbot), and

result (overall game score);

- user demographics — for each user: *age*, *location* (U.S. state or country), *native language* and *gender*[5];
- opinion poll — users are asked to rank their experiences (optionally) whenever they exit a playground: on a sliding scale of *easy* to *hard* and *boring* to *exciting*;
- email from users; and
- interviews with students and teachers at our test site.

Three basic measurements will be used for the human data:

- *rate of return* — how frequently a player returns to the CEL web site and how many games s/he plays during a visit;
- *rate of success* — the number of wins and losses incurred by a player; and
- *domain coverage* — the amount of the knowledge domain that a player has experienced (i.e. been exposed to).

Analysis will show if there is any correlation amongst these values, particularly between:

- rate of return and rate of success;
- rate of return and domain coverage;
- rate of success and domain coverage;
- rate of return, rate of success, domain coverage and user demographics;
- rate of return, rate of success, domain coverage and opinion poll; and
- rate of return, rate of success, domain coverage and direct user feedback (email and interviews).

Human learning studies will center around comparing the following elements (see figure 4): player anonymity (either *known* or *anonymous*), playmate choice (either users have a *choice* of who to play against, or they have *no choice* and the system arranges matches), and playgroup composition (either *open*, meaning that players of all levels are placed in the same playgroup, or *restricted*, where players are clustered appropriately).

[5] Users are asked to provide this information the first time they log into CEL. Supplying this data is optional and its value is dubious, since it may be extremely noisy. However, standards are emerging for analyzing these types of information statistically, and it may be possible to glean some measure of honesty by comparing entries for *location* with IP addresses. The purpose here is to see if any trends in return rate and/or success rate correlate to user demographics.

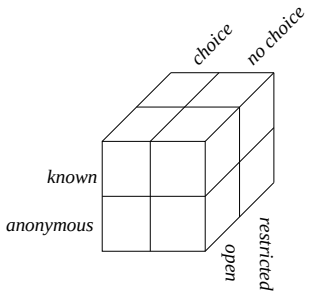


Figure 4: Human learning experiments.

In addition, the algorithms used for player clustering will be compared. Studies will examine:

- using absolute rank versus relative distance between players as the basis for forming playgroups, and
- varying the value of ϵ in the previous calculations.

Machine learning studies will focus on the adaptation and success of the two types of agents. Various methods for bitbot selection will compare:

- using different genetic reproduction operators to control the composition of a problem set,
- varying the mixture of easy and hard bitbots in a problem set, and
- trying several methods for building bitbot sets for a pair of users – using only bitbots selected particularly for the "better" player or for the "slower" player, or some combination of the two.

For these trials, one method for all players and different methods for different players may be tried, based on a players' rate of return and rate of success.

The effectiveness of the secret agents will be examined, to determine if the agents successfully fool their human counterparts and if they have become viable learning partners in the system. Analysis will compare game results for human-human versus human-agent matches.

Finally, the notion of self-organizing curriculum (mentioned in section 6.1) will be explored. If an ordering can be found in the aggregate human

data, those "stages" will be compared with standard curriculum. This may give insight into whether using the co-evolutionary method to discover paths within a curriculum area has merit.

8 Thesis outline

The preliminary outline of the thesis contains the following chapters:

1. Introduction.
2. System architecture.
3. Problem selection.
4. Student modeling.
5. Player clustering.
6. Secret agents.
7. Conclusions and future work.

Reviews of related work and results of experiments will be incorporated within chapters 3-6, according to the subject of each chapter.

9 Schedule

Currently, the initial CEL system is up and running with one game (Keyit). We have begun taking data from 4th and 5th grade children, in a controlled computer laboratory setting at the Maria Hastings Elementary School in Lexington, Massachusetts.

A tentative schedule for the work is as follows:

15 Mar 1999	Gave proposal to Jordan
26 Mar 1999	Began co-evolution of bitbots
6 Apr 1999	Released simple secret agents
1 May 1999	Release arithmetic game
15 May 1999	Implement initial player clustering algorithm
1 Jun 1999	Post CEL link on DEMO home page, Tron and HC-gammon pages Begin preliminary analysis of bitbot data to determine if learning is being detected and tracked
15 Jun 1999	Set up eight human experiments to run during the summer
1 Jul 1999	Complete chapter 2
1 Aug 1999	Complete clustering literature survey (ch 5) Complete student modeling literature survey (ch 4)
15 Aug 1999	Analyze student modeling experiments (ch 4) (preliminary)
1 Sep 1999	Analyze player clustering experiments (ch 5) (preliminary)
15 Sep 1999	Complete chapter 4 (add final data analysis in January)
1 Oct 1999	Complete chapter 5 (add final data analysis in January)
15 Oct 1999	Complete evolutionary computation literature survey (ch 3) Complete neural network literature survey (ch 6)
1 Nov 1999	Analyze bitbot experiments (ch 3) Analyze secret agent experiments (ch 6)
15 Nov 1999	Complete chapters 3 and 6 (add final data analysis in January)
15 Dec 1999	Complete chapters 1 and 7
15 Jan 2000	Defense

References

[1] M.E. Shiri A., E. Aïmeur, and C. Frasson. Case-based student modelling: unaccessible solution mode. In *Conference internationale sur les nouvelles technologies de la communication et de la formation (NTICF'98)*, 1998.

[2] J. R. Anderson. Acquisition of cognitive skill. *Psychology Review*, 89, 1982.

[3] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Genetic Algorithms: Proc. 5th Int'l Conference (GA93)*, 1993.

[4] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[5] M. Balabanović. *Learning to Surf: Multiagent Systems for Adaptive Web Page Recomendation*. PhD thesis, Stanford University, 1998.

[6] J. Beck. Modeling the student with reinforcement learning. In *Proceedings of the Machine Learning for User Modeling Workshop, Sixth International Conference on User Modeling*, 1997.

[7] R.D. Beer. Toward the evolution of dynamical neural networks for minimally cognitive behavior. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 421–429, 1996.

[8] P. Bell, E.A. Davis, and M.C. Linn. The knowledge integration environment: Theory and design. In *Proceedings of Computer Supported Collaborative Learning (CSCL'95)*, 1995.

[9] H.J. Berliner. Backgammon computer program beats world champion. *Artificial Intelligence*, 14, 1980.

[10] A. D. Blair and J.B. Pollack. What makes a good co-evolutionary learning environment? *Australian Journal of Intelligent Information Processing Systems*, 4(3/4):166–175, 1997.

[11] J. S. Brown and R. B. Burton. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2(2), 1978.

[12] A. Bruckman. *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual Community for Kids*. PhD thesis, MIT, 1997.

[13] P. Brusilovsky, E. Schwarz, and G. Weber. Elm-art: An intelligent tutoring system on world wide web. In C. Frasson, G. Gauthier, and A. Lesgold, editors, *Intelligent Tutoring Systems (Lecture Notes in Computer Science, Vol. 1086)*, pages 261–269. Springer Verlag, 1996.

[14] W. J. Clancey. Intelligent tutoring systems: A tutorial survey. Technical Report STAN-CS-87-1174, Stanford University, 1986.

[15] D. Cliff and G.F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In *Proceedings of the European Conference on Artificial Life (ECAL'95)*, 1995.

[16] C. Conati and K. VanLehn. Pola: a student modeling framework for probabilistic on-line assessment of problem solving performance. In *Proc. 5th Int'l Conf. User Modeling (UM-96)*, 1996.

[17] E. De Corte. Learning theory and instructional science. In P. Reimann and H. Spada, editors, *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*. Pergamon, 1995.

[18] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for InformationScience*, 41(6):391–407, 1990.

[19] B.L. Digney. Learning and shaping in emergent hierarchical control systems. In *Proceedings of Space'96 and Robots for Challenging Environments II*, 1996.

[20] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71:321–370, 1994.

[21] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[22] S.L. Epstein. Toward an ideal trainer. *Machine Learning*, 15(3):251–277, 1994.

[23] T. Fanderclai. Muds in education: New environments, new pedagogies. *Computer-Mediated Communication Magazine*, 2(1), 1995.

[24] S.G. Ficici and J.B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proc. of ALIFE-6*, 1998.

[25] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[26] L.J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[27] L. Foner. Yenta: A multi-agent referral based matchmaking system. In *Proceedings of the First International Conference on Autonomous Agents (Agents97)*, 1997.

[28] J. Forrester. System dynamics and learner-centered-learning in kindergarten through 12th grade education. Technical Report D-4337, MIT, 1992.

[29] P. Funes, E. Sklar, H. Juillé, and J. Pollack. Animal-animat coevolution: Using the animal population as fitness function. In *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, 1998.

[30] P. Funes, E. Sklar, H. Juillé, and J.B. Pollack. The internet as a virtual ecology: Coevolutionary arms races between human and artificial populations. Computer Science Technical Report CS-97-197, Brandeis University, 1997.

[31] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave and information tapestry. *Communications of the ACM*, 35(12), 1992.

[32] I. Goldstein. Developing a computational representation for problem solving skills. In *Proc. Carnegie-Mellon Conf. Problem Solving and Education: Issues in Teaching and Research*, 1978.

[33] A. Gordon and L. Hall. Collaboration with agents in a virtual world. In *Workshop on Current Trends and Applications of Artificial Intelligence in Education: 4th World Congress on Expert Systems*, 1998.

[34] H. E. Gruber and J. J. Voneche, editors. *The Essential Piaget*. BasicBooks, 1977.

[35] T. Haynes, R. Wainwright, S. Sen, and D. Schoenefeld. Strongly typed genetic programming in evolving cooperative strategies. In L. Eshelman, editor, *Genetic Algorithms: Proc. of 6th International Conference (ICGA95)*, 1995.

[36] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton et al., editor, *Proc. of ALIFE-2*, pages 313–324. Addison Wesley, 1992.

[37] J. H. Holland. *Adaption in Natural and Artificial Systems*. Univ. of Michigan Press, 1975.

[38] S. Jackson, J. Krajcik, and E. Soloway. The design of guided learner-adaptable scaffolding in interactive learning environments. In *Proceedings of the Human Factors in Computing Systems Conference (CHI98)*, 1998.

[39] D.W. Johnson and A. Ahlgren. Relationship between student attitudes about cooperation and competition and attitutdes towards schooling. *Journal of Educational Psychology*, 68(1):92–102, 1976.

[40] D.W. Johnson and R. Johnson. Cooperative learning, values, and culturally plural classrooms. In *Cooperation and competition: Theory and research*. Interaction Book Company, 1989.

[41] H. Juillé and J. B. Pollack. Dynamics of co-evolutionary learning. In *Proc. of SAB-4*, pages 526–534. MIT Press, 1996.

[42] K. Koedinger and J. Anderson. Effective use of intelligent software in high school math classrooms. In *Proceedings of the World Conference on Artificial Intelligence in Education*, 1993.

[43] A. Kohn. *No Contest: The case against competition*. Houghton-Mifflin, 1986.

[44] J.L. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7, 1983.

[45] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[46] D. Kuokka and L. Harada. Matchmaking for information agents. In M.N. Huhns and M.P. Singh, editors, *Readings in Agents*. Morgan Kaufman, 1997.

[47] K. Lang. Newsweeder: Learning to filter news. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[48] P. Langley. Order effects in incremental learning. In P. Reimann and H. Spada, editors, *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*. Pergamon, 1995.

[49] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.

[50] M. Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2:103–138, 1987.

[51] W-P. Lee, J. Hallam, and H.H. Lund. A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proc. 1996 IEEE Int'l Conf. Evolutionary Computation*, pages 384–389, 1996.

[52] H. Leiberman. Letizia: An agent that assists web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995.

[53] P. Maes. How to do the right thing. *Connection Science Journal*, 1(3):291–323, 1989.

[54] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40,146, 1994.

[55] P. Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1-2):135–162, 1994.

[56] G.I. McCalla and J.E. Greer. Granularity-based reasoning and belief revision in student models. In *Student Models: The Key to Individualized Educational Systems*, pages 39–62, New York, 1994. Springer Verlag.

[57] D. Michie. Trial and error. *Science Survey*, part 2:129–145, 1961.

[58] G.F. Miller and D. Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 411–420, 1994.

[59] S. Milne, E. Shiu, and J. Cook. Development of a model of user attributes and its implementation within an adaptive tutoring system. *User Modeling and User-Adapted Interaction*, 6(4), 1996.

[60] A. Mitrovic, S. Djordjevic, and L. Stoimenov. Instruct: Modeling students by asking questions. *User Modeling and User-Adapted Interaction*, 6(4), 1996.

[61] J. Mostow and G. Aist. The sounds of silence: Towards automated evaluation of student learning in a reading tutor that listens. In *Proc. AAAI-97*, 1997.

[62] K. Nakabayashi, M. Maruyama, Y. Koike, Y. Kato, H. Touhei, and Y. Fukuhara. Architecture of an intelligent tutoring system on the www. In *Proc. of AIED-97*, 1997.

[63] L. Owens. An international comparison of the learning preferences of secondary students: Australia and england. In *Proceedings of the Annual Conference of the Australian Association for Research in Education*, 1991.

[64] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas.* BasicBooks, 1980.

[65] S. Papert. *The Children's Machine.* BasicBooks, 1993.

[66] R. Pea. The collaborative visualization project. *Communications of the ACM*, 36(5):60–63, 1993.

[67] S. Perkins and G. Hayes. Incremental acquisition of complex behaviour using structured evolution. In G.D. Smith, N.C.Steele, and K.F. Albrecht, editors, *Proceedings of International Conference on Neural Networks and Genetic Algorithms.* Springer-Verlag Wein, 1997.

[68] J. B. Pollack, A. D. Blair, and M. Land. Coevolution of a backgammon player. In C. Langton, editor, *Proc. of ALIFE-5.* MIT Press, 1996.

[69] J.B. Pollack and A.D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32:225–240, 1998.

[70] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[71] A. Qureshi. Evolving agents. In *Proceedings of the First International Conference of Genetic Programming (GP-96)*, 1996.

[72] M. Resnick. *Turtles, termites, and traffic jams: explorations in massively parallel microworlds.* MIT Press, 1997.

[73] C. W. Reynolds. Competition, coevolution and the game of tag. In R. A. Brooks and P. Maes, editors, *Proc. of 4th Int'l Workshop on the Synthesis and Simulation of Living Systems.* MIT Press, 1994.

[74] Elaine Rich. *Artificial Intelligence.* McGraw-Hill, 1983.

[75] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229, 1959.

[76] G. Saunders, J.F. Kolen, and J.B. Pollack. The importance of leaky levels for behaviour-based ai. In *Proceedings of Third International Conference on Simulation of Adaptive Behavior*, 1994.

[77] R. Schank and C. Cleary. *Engines for Education.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1995.

[78] R. C. Schank. Failure-driven memory. *Cognition and Brain Theory*, 4(1), 1981.

[79] K. Sims. Evolving 3d morphology and behavior by competition. *Proc. Artificial Life 4*, pages 28–39, 1995.

[80] E. Sklar, A. Blair, and J. Pollack. Co-evolutionary learning: Machines and humans schooling together. In *Workshop on Current Trends and Applications of Artificial Intelligence in Education: 4th World Congress on Expert Systems*, 1998.

[81] E. Sklar and J.B. Pollack. Toward a community of evolving learners. In *Proceedings of the Third International Conference on the Learning Sciences (ICLS-98)*, 1998.

[82] E. Sklar and J.B. Pollack. The design of the cel system. Computer science technical report, Brandeis University, *(in progress)*.

[83] R.E. Slavin. When and why does cooperative learning increase achievement? theoretical and empirical perspectives. In R. Hertz-Lazarowitz and N. Miller, editors, *Interaction in cooperative groups: The theoretical anatomy of group learning*, pages 145–173. Cambridge University Press, 1992.

[84] R.E. Slavin. *Cooperative Learning: Theory, Research, and Practice*. Allyn & Bacon, 1995.

[85] E. M. Soloway, B. Woolf, E. Rubin, and P. Barth. Meno-ii: An intelligent tutoring system for novice programmers. In *Proc. 7th Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 1981.

[86] M. Stern, B. Woolf, and J.F. Kurose. Intelligence on the web? In *Proceedings of the 8th World Conference of the AIED Society (AIED'97)*, 1997.

[87] D. Suthers and D. Jones. An architecture for intelligent collaborative educational systems. In *Proceedings of the 8th World Conference of the AIED Society (AIED'97)*, 1997.

[88] D. Suthers, A. Weiner, J. Connelly, and M. Paolucci. Belvedere: Engaging students in critical discussion of science and public policy issues. In *Proc. of AIED-95*, 1995.

[89] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning 3*, 1988.

[90] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8, 1992.

[91] G. Tesauro. Temporal difference learning and td-gammon. *Commun. of the ACM 39(3)*, 1995.

[92] M. Turoff. Designing a virtual classroom [tm]. In *Proceedings of the 1995 International Conference on Computer Assisted Instruction (ICCAI'95)*, 1995.

[93] K. VanLehn. Human procedural skill acquisition: Theory, model, and psychological validation. In *Proceedings of the National Conference on AI*, 1983.

[94] K. VanLehn, Z. Niu, S. Slier, and A. Gertner. Student modeling from conventional test data: A bayesian approach without priors. In *Proceedings of the 4th Intelligent Tutoring Systems Conference (ITS'98)*, pages 434–443, 1998.

[95] L.S. Vygotsky. *Mind in society : the development of higher psychological processes*. Harvard University Press, Cambridge, 1978.

[96] J. Walters and B. Hughes. Camp marimuse: Linking elementary and college students in virtual space. In *Proceedings of the National Educational Computing Conference*, 1994.

[97] U. Wilensky. Making sense of probability through paradox and programming. In *Constructionism in Practice: Designing, Thinking and Learning in a Digital World*. Erlbaum, 1996.