Hindawi Publishing Corporation
International Journal of Computer Games Technology
Volume 2009, Article ID 456169, 9 pages
doi:10.1155/2009/456169

Research Article

Enhancing Artificial Intelligence on a Real Mobile Game

Fabio Aiolli and Claudio E. Palazzi

Department of Pure and Applied Mathematics, University of Padova, Via Trieste 63, 35131 Padova, Italy

Correspondence should be addressed to Claudio E. Palazzi, cpalazzi@math.unipd.it

Received 30 May 2008; Accepted 8 September 2008

Recommended by Kok Wai Wong

Mobile games represent a killer application that is attracting millions of subscribers worldwide. One of the aspects crucial to the commercial success of a game is ensuring an appropriately challenging artificial intelligence (AI) algorithm against which to play. However, creating this component is particularly complex as classic search AI algorithms cannot be employed by limited devices such as mobile phones or, even on more powerful computers, when considering imperfect information games (i.e., games in which participants do not a complete knowledge of the game state at any moment). In this paper, we propose to solve this issue by resorting to a machine learning algorithm which uses profiling functionalities in order to infer the missing information, thus making the AI able to efficiently adapt its strategies to the human opponent. We studied a simple and computationally light machine learning method that can be employed with success, enabling AI improvements for imperfect information games even on mobile phones. We created a mobile phone-based version of a game called *Ghosts* and present results which clearly show the ability of our algorithm to quickly improve its own predictive performance as far as the number of games against the same human opponent increases.

1. Introduction

Mobile phones have brought into our lives the possibility and the willingness to be always reachable by anybody; they have almost become an extension of ourselves, making us *Homo Mobilis* for how much we tend to never separate from them [1]. Service providers are riding this wave by continuously offering new appealing services. Among the others, mobile games represent a great and ever-increasing source of revenue in the mobile service market. Indeed, market studies report incomes for the wireless gaming industry in the order of billions of US dollars worldwide and with a 40% growth each year [2–4].

Gaming has always been one of people's favorite digital applications. Nowadays, three main reasons have enabled its success even in the mobile market [5]. First, the incredible proliferation of mobile phones, which have surpassed in number base line phones in countries such as Finland and Italy, thus creating hundreds of millions of potential customers. Second, technological advances have transformed mobile phones from a cordless version of a regular phone into a hand-held computer able to deliver quality audio/video and quickly run complex algorithms, as those required by recent games. Third, the increasing availability of wireless connectivity (i.e., GPRS, UMTS, Bluetooth, Wi-Fi) provides the possibility to play online with other people and allows to create new business models where the game is bought online and directly downloaded in the mobile phone.

While the trend toward a massive use of mobile games is out of dispute, several technical problems remain unsolved; the market success of future (and present) mobile games also passes through providing valid answers to them. For instance, a challenging research issue is related to the availability of an adequate AI algorithm. Indeed, users often play alone against the AI; this could happen to avoid the connectivity cost to play against another mobile user, or because a mobile phone may have gaming capabilities but no connectivity, or just because the user prefers so. Therefore, the game has to be endowed with an AI that is fun to play against: neither too trivial, nor too tough. Unfortunately, classic searching techniques may not be feasible for the considered context for two main reasons: (i) a game could be played on mobile phones with limited computational power, (ii) these techniques might not function when considering certain games. Elaborating on this second point, we have to remember that games can be classified into two main categories, *perfect information games* (e.g., Chess) and *imperfect information games* (e.g., Poker), depending on whether participants have or do not have a complete knowledge of the game state at any moment. When players have just a very limited knowledge of the game state, resorting to traditional state space searches, may result in an AI which behaves similarly to a random decision maker.

As a case study for this problem, we have created a mobile phone-based version of an imperfect information game named *Ghosts,* a simple board game played by two opponents. The board game has been invented by Alex Randolph and is sold in Germany by Drei Magier Spiele. Its original (German) name is: "*Die guten und die bösen Geister,*" that is, "good and bad ghosts;" for brevity, we simply name it *Ghosts*. The game is particularly interesting for our study as players do not have a complete knowledge of the game state: they can both see the position of game pieces on the board, but they cannot see the type of the opponent's ones. Depending on this information, different tactics will be adopted (i.e., attack the opponent's piece, leave it alone, run away from it). Therefore, in order to win, a player has also to infer the type of each of the opponent's pieces. This information can be extracted from the player's behavior, also keeping in mind that different players can adopt different strategies, for instance, by resorting more or less frequently to bluffing.

To this aim, as main research contribution of this work, we have developed a new kind of AI solution that is able to adapt the gaming strategy to the current human player inferring unknown information about the game state. Our solution employs machine learning techniques to mimic the human's ability in intuiting the opponent's intentions after several game sessions and is hence particularly helpful with imperfect information games. Simply, it associates the behavior features of a player with a presumed type of a piece; by observing how a player acts in different game state configurations, the AI becomes able to classify tactics employed by that player and to adapt to them.

Even if this AI solution represents our main scientific contribution, while creating our mobile version of the game *Ghosts*, we have not overlooked at two practical problems that are crucial in the successful deployment of a real mobile game: (i) compatibility with the highest number of mobile phones in the market and (ii) connectivity among players' mobile phones [5]. To this aim, we have evaluated possible alternatives and finally adopted the state-of-the-art technology that allows the widest compatibility and connectivity among existing mobile phones.

The rest of the paper is organized as follows. Since we developed a real mobile game with an original AI solution, in Section 2, we discuss development issues that are at the basis of implementative choices and, in Section 3, we overview related AI scientific works. Then, in Section 4, we technically present our developed mobile game. As our novel AI approach represents the main research contribution of this work, we devote Section 5 to discuss its details. Section 6 describes the experimental scenario and reports the corresponding outcomes. Finally, in Section 7, we provide a conclusion and future directions for this work.

2. Technical Development Background

In this section we provide background information about two technical issues which are very important to take into account in order to develop a successful mobile game, namely, compatibility and connectivity.

2.1. Compatibility. The current mobile gaming panorama is affected by a significant fragmentation problem that precludes making a game available to the entire mobile market [6]. This is caused by both the absence of a standard in terms of software platform for mobile phones [5] and by the specific characteristics (e.g., screen size) of different phones, even when produced by the same manufacturer. Indeed, if game designers wanted to fully exploit the features of a given device, they should renounce to have that game also running on entry-level phones. Another practiced solution is that of developing multiple versions of the same game to have one version specifically designed for each class of mobile phone; clearly this solution has a cost.

Currently, we can identify three software platforms that emerge as the most popular ones when considering mobile games with connectivity capabilities: Symbian [7], Binary Runtime Environment for Wireless (BREW) [8], and Java Micro Edition (Java ME) [9]. The first one is a proprietary operating system that has been developed by a consortium among Nokia, Sony Ericsson, Siemens, Panasonic, and Samsung. As these brands represent a very wide portion of the global mobile market, Symbian can be considered a very popular operating system. Symbian applications have also the advantage of being fast as they are generally written in C++ and can make use of specific features of the considered mobile phone. This may require specific programming skills and raises compatibility issues with other software platforms.

BREW is a development platform for mobile phones and it is based on C++. As a main advantage, BREW is located between the application layer and the operating system of the mobile phone; this way, it offers a simple interface to the programmer to handle different system/networking details. Unfortunately, BREW is not a free platform and this significantly limits its popularity.

Finally, Java ME is a Java application environment designed for devices with limited capabilities in terms of processing power, memory, and graphics. Its Connected Limited Device Configuration (CLDC) is composed by a set of libraries specifically designed to run Java applications on limited devices; this set of libraries is extended by the Mobile Information Device Profile (MIDP) that embodies a set of APIs for the GUI, the data storage, and the networking functionalities. The latest release, MIDP 2.0, provides specific APIs also to generate 3D graphics for games. Mobile applications written in Java ME are named MIDlet and, although they do not run as fast as applications purposely designed for a particular device or platform, their main advantage is that, potentially, they can be run on any Java ME-compatible device.

All these platforms have both pros and cons, thus demonstrating the need for a solution that will enable the automatic porting of any mobile game on any mobile phone. However, this is not the aim of this work; we simply note that currently Java ME with MIDP 2.0 is the solution that provides the widest portability of the developed software.

2.2. Connectivity. Different communication technologies are available today on most of mobile phones (e.g., GPRS, UMTS, Bluetooth, Wi-Fi); thereby, being able to exploit them has become an important aspect in the success of a mobile game [5]. The current mobile scenario is dominated by 2G and 3G (GPRS and UMTS, resp.). Phone service providers have done huge investments on this technology; therefore, this communication means present the advantage of being available almost anywhere. Yet, its bandwidth, latency, and cost often block users from using it.

Bluetooth connectivity is also very popular today as only really cheap mobile phones are produced today without it. Bluetooth was designed to implement personal area networks (PANs) and, thereby, its bandwidth and latency also allow to support multimedia applications [10]. Transmissions happen only in a $\sim$10 m range, which implies that players have to be one in front of (or beside) the other to play together; yet, this proximity in the real world is often part of the fun of playing together. Finally, Bluetooth communications are not billed.

Wi-Fi is another communication technology that can be free of charge (or available at a low fixed cost). Its transmission range is in the order of 100 m but can be used as well to play online with other people all around the world by simply connecting to the Internet through an access point in proximity [11]. Unfortunately, Wi-Fi capabilities are currently present only on expensive mobile phones; plus, while walking in a street, there might not be around any freely accessible Wi-Fi access point thus impeding its use.

The optimal solution would be that of having the game enabled to work on any of the aforementioned connectivity means and choosing, at any moment, the "best" among the available ones (e.g., the fastest, the cheapest, the most reliable) [12, 13]. However, if a mobile game producer decides to create a game with only one connectivity option, we deem that the chosen one should be Bluetooth as it is available on almost any new mobile phone and its use is free of charge. The combination of these two characteristics makes users willing to use it for their leisure.

3. Artificial Intelligence Related Work

In this paper, we present a novel AI approach for imperfect information games; we hence deem important to devote this section to provide a discussion about related work in AI.

The first self-learning gaming program, that is, Checkers, was created in 1959 and represented a very early demonstration of the fundamental concept of AI in games [14]. Nowadays, all video games include some AI that may act as a virtual opponent or as a component of the game itself. Yet, the AI of current games show only little advancements if compared to its ancestors; only for few specific games the AI has achieved great improvements (e.g., Chess [15]).

As already stated in Section 1, games can be categorized into two main classes: games where the players possess perfect information about the current game state (e.g., Chess, Tic-tac-toe) and games where players can rely on imperfect information only (e.g., Poker, Rock-paper-scissors). The AI of perfect information games can easily evaluate a given game state by just searching all possible continuations to a fixed depth. For this kind of games, the main problem in developing an AI is related to the capability of precomputing correct evaluations of each game state and then storing and retrieving them in an efficient manner [16–18].

Instead, with imperfect information games, deep search may not be feasible and storing precomputed evaluations might not result in significant improvements in the AI's strength [19, 20]. In this case, techniques like temporal difference learning are also unsuitable as the intermediate states of a game are only partially determined [21]. Alternative solutions have hence to be explored to enhance the level of the AI. For instance, simulation search [22–25] evaluates the possible next moves by self-playing a multitude of simulated game sessions, considering the current state as the starting point and utilizing different values for the indeterministic parameters (i.e., dice rolls, cards held by the opponent player, etc.). To generate real-time responses during the game, these simulations can be run before the game and statistics can be stored to be promptly available during the game. Unfortunately, the branching factor of certain games may considerably limit the effectiveness of this technique.

To this aim, in Section 5 we discuss a new machine learning approach that mimics the human's ability in evaluating important information about the current game state that goes beyond, for instance, the board position [26]. In essence, our mechanism models the opponent's behavior over several game sessions so as to be able to exploit the weaknesses and the typical behaviors of the considered human player.

4. A Representative Case of Imperfect Information Game: *Ghosts*

For our study, we need a simple, yet representative exemplar of imperfect information game. *Ghosts* embodies a perfect case as, not only it belongs to this class of games, but it is also governed by few simple rules, which makes it easier to appreciate the improvement produced by the employment of our AI solution.

The rules of *Ghosts* are listed hereinafter. Two players have to place 8 ghosts each at the back of a 6×6 board as shown in Figure 1. Each player has 4 good ghosts and 4 bad ghosts, but the information about which are good and which are bad is hidden to the opponent player. On each turn, a player moves one of her/his ghosts one square vertically or horizontally; if by doing so the ghost is moved onto an opponent's ghost, the latter is captured by the former. In

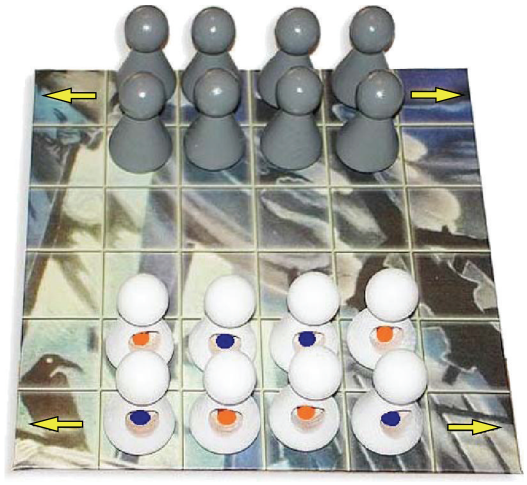
Figure 1: Initial setup of *Ghosts*.

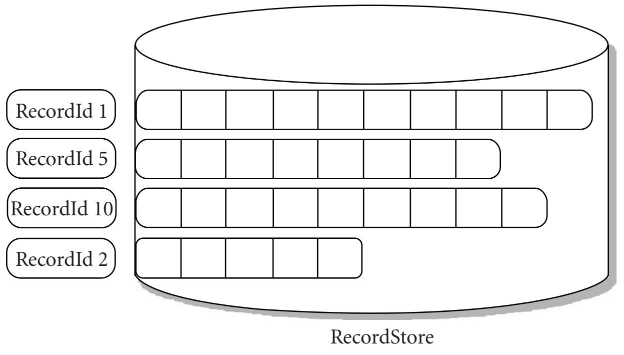

Figure 2: The record management system.

order to win, different possibilities are available to a player: (i) having all of her/his bad ghosts captured by the opponent player, (ii) capturing all the good ghosts of the opponent player, (iii) moving one of her/his good ghosts off the board from one of the opponent's corner squares. Clearly, one of the interesting aspects of *Ghosts* is its bluffing element which is differently utilized by different human players.

4.1. Development of Ghosts for Mobile Phones. The development of the digital version of *Ghosts* for mobile phones has passed through three different phases: creating the GUI and the basic logic of the game, enabling communication among two mobile phones, adding the AI with our player profiling capability. We discuss in this section the first two of these phases, which are related to the practical implementation of the game; we leave the third one to the next section as its research contribution deserves a deeper and different kind of discussion.

To create the game, we decided to use Java ME. This choice was motivated by several reasons. First, developing mobile games through the APIs of MIDP 2.0 is quite simple. Second, Java ME comes also with an emulator that facilitates software development before trying it on real mobile devices. Third, we wanted to create a game with the highest portability on different devices; indeed, we tested our final game on various mobile phones (Nokia 7610, 6630, 6670, 6260, N70, N95) and it perfectly worked in all cases.

Unfortunately, memorizing data related to a mobile game cannot be done in a trivial way as we would do with regular computer-based games. The problem is related to the employed mobile device; indeed, in our phone, we had to resort to a specific data structure named Record Management System (RMS). More in detail, Figure 2 shows that RMS is modeled on the basis of a simple database where stored data are organized in records.

The connectivity of the game has been ensured through Bluetooth. We are currently adding the possibility to exploit Wi-Fi and we plan to enable GPRS and UMTS in the future; however, having to choose just one connectivity means to start with, we preferred Bluetooth since it is available on most of the mobile phones on the market, its use is free of charge, and it does not require networking infrastructure (i.e., access points, game servers) to work.

With Bluetooth connectivity, one of the involved devices has to act as a server and the other(s) can be client(s). In our game, which phone acts as a server and which one as a client is explicitly chosen by the two players when starting the game session. With the help of Figure 3 we describe the various phases to initiate a game session among two mobile phones. First, our *Ghosts* application has to be started on both mobile phones to enable the local device. A choice is made between activating the game as a server or as a client; then, in the former case, the service is created and waits for a client, whereas in the latter case, the client has to search for devices within its transmission range. Once a device is found, the client searches for a specific service (i.e., the game server) on that device; if the service is discovered then the connection is established and messages (e.g., game moves) can be transmitted in turn from one device to the other in order to play the game (our game prototype running on a mobile phone is shown in Figure 4).

5. Enhancing the AI with Player Profiling Capabilities

The last component of our mobile game is represented by the AI algorithm that allows players to play in solo mode; it is also the most interesting from a research point of view.

Problems in game AI are typically grouped with respect to certain characteristics. One of the most intriguing is related to how much information is available to the players. This leads us to distinguish between perfect information games and imperfect information games. The latter is more interesting as it represents a more challenging case.

Ghosts embodies a good case study as it falls in the class of imperfect information games, yet, it is simple enough to be analyzed. In *Ghosts*, a player does not have any information about the type of the opponent's ghosts (i.e., good or bad); thereby, any search state space-based technique, for example, min-max algorithms, will fail. In other words, without any other heuristic judgment, the behavior of a machine driven player could not be better than any trivial random player.

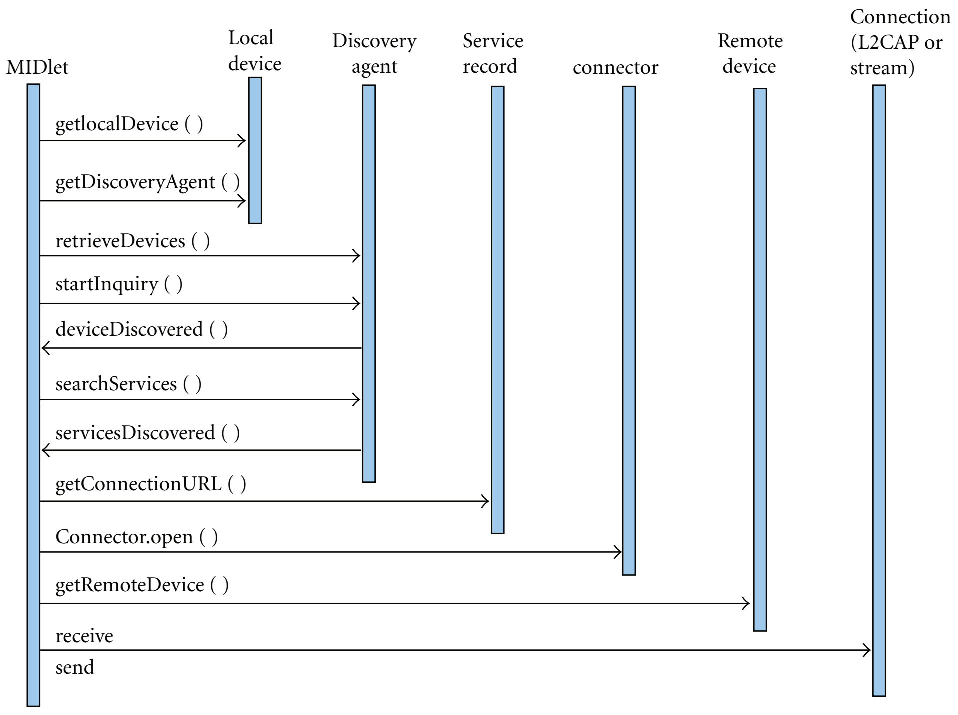

Figure 3: Connection steps through Bluetooth.

Figure 4: Our final *Ghosts* game running on a mobile phone.

Indeed, in order to plan its moves, an AI algorithm would certainly benefit from some other source of information about the type of the opponent's ghosts (the missing information). We propose to get this additional information from the playing style of a player. The basic assumption we make is that different players have different playing behaviors (being aggressive, bluffing, etc.) and they tend to move pieces of a certain type in a similar way when facing similar game situations. Specifically, our claim is that by knowing the playing style of a player, it is possible to recognize the type of a given ghost by its moves. We can then use a machine learning algorithm [21] to compile a behavior profile of good and bad pieces of a player. During future game sessions, this knowledge can be used to predict the type of a ghost in the board and possibly to define higher level game heuristics, like min-max algorithms, based on these predictions. This kind of prediction can be easily seen as a classification problem in machine learning, as discussed in the following section.

5.1. Machine Learning for Classification. One of the most important problems in machine learning is classification. In this setting, given a set of preclassified instances, one wants to predict the class of new instances never seen before. In the simplest case we have two labels, say $\{-1, +1\}$, denoting that an instance belongs (+1), or not belongs (−1), to a given concept (or class). Instances of the problem are described by d-dimensional vectors, each component containing the value of a given attribute (feature) relative to the instance. In general, it is assumed that the instance-class pairs are drawn according to a fixed (but unknown) probability distribution. Moreover, the set of preclassified examples, that is, the training set, is assumed to be drawn according to the same probability distribution.

Various algorithms have been designed to solve this problem; they can be grouped into two big families: discriminative and nondiscriminative methods. Given any class, a nondiscriminative method tries to estimate the probability of an instance using the training examples; then, it uses the Bayes rule (or similar, [27]) to estimate the probability of each class given an instance; finally, through these estimates, any new instance can be classified with the most likely class. Note that each class is treated independently and the estimation of the probability of instances given the class can be done by using instances of that class only.

A discriminative method tries to directly estimate the posterior probability of each class given the instance. For this last estimate, all the examples of the training set have to be

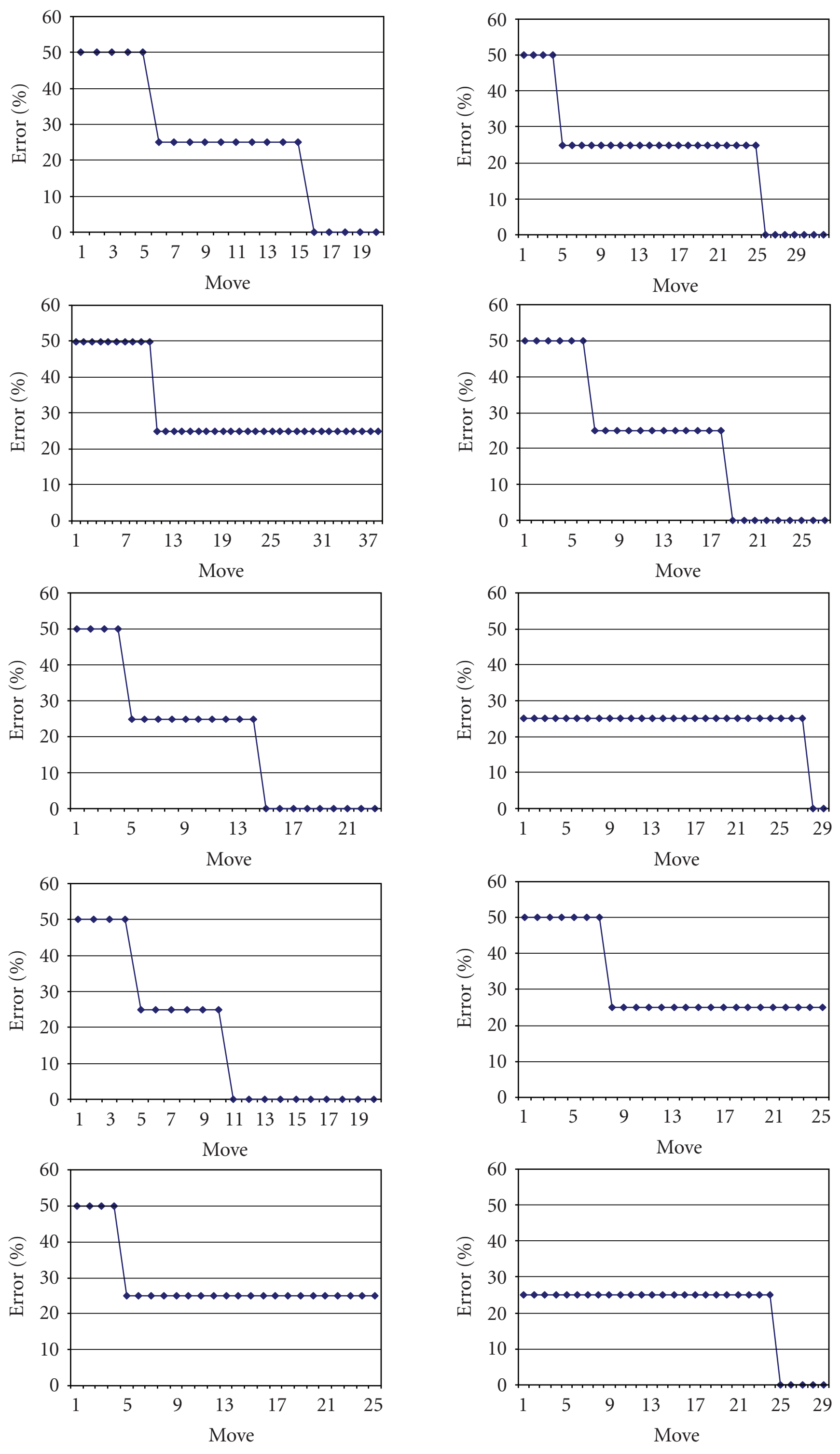

Figure 5: Percentage of error of the prototype-based machine learning algorithm (proportion of times a type of a piece on the board is wrongly predicted) for 10 different game sessions. The *x*-axis represents the moves of the player during that game session (the number of moves for a game varies on different game sessions).

used together. Generally, a nondiscriminative method tends to be more efficient and computationally less demanding than discriminative methods. However, discriminative methods usually have best performances.

One of the most successful discriminative classification methods is the Support Vector Machine (SVM) [28]. With this method, instances are mapped into a very large feature space where a linear separator (a hyperplane) is found by the learning algorithm. Advantages of SVM with respect to previously devised methods, like neural networks, for example, are the solid theoretical framework on which they are defined and the small number of hyperparameters that a user has to tune to use them. Furthermore, since the SVM solution can be found by optimizing the dual of a convex constrained quadratic objective function, this solution does not suffer of local minima as neural networks do. Moreover, the solution can be expressed in the very simple form of a linear combination of functions which depend on training instances in the original (lower dimensional) space.

The problem with SVM is that, since the generated model is defined on the basis of a subset of training examples, the evaluation of the decision function can be onerous when the training set size is big. This makes SVM quite unsuitable when a fast reaction of the classifying system is expected (e.g., in a real-time application) or when the utilized device has limited computational capabilities. Furthermore, the storage memory required by the model generated by an SVM is of an order linear with the cardinality of the training set, and this could be an issue when devices with a very limited memory should be used (e.g., mobile devices).

Among nondiscriminative classification methods, prototype-based methods are probably the easiest and less computationally demanding. In this method, a prototype is built for each class representing the common patterns of instances of the associated class. Specifically, once given a distance metric between instances, the basic idea is to build a prototype vector for each class, having the same dimension of the instances, in such a way to minimize the mean value of the distances between the prototype and each of the instances of the class. Once these prototypes are built, a new instance is classified with the class associated with the closest prototype. This method is theoretically founded when some statistical assumptions are made about the distribution of the examples and is empirically demonstrated to obtain good performances.

5.2. Prototype-Based Classification in Ghosts . Since our particular application is thought to work even with very limited computational resources, we have adopted a prototype-based classifier as the machine learning methodology. Specifically, for each player, a prototype of good and a prototype of bad ghost behavior are trained based on 17 features which have been considered informative to determine the nature of a ghost in the game.

In particular, the following features have been chosen: 8 features with binary values representing what was the initial position of the piece on the board among the eight possible ones, 5 features representing the moves of the piece during the game session (if it is the first piece that the player moved, if it is the second piece that the player moved, the number of backward, forward, and lateral moves already performed by that piece), and the remaining 4 features representing the piece's behavior when it has been under threat of being captured (how many times it has reacted by capturing the opponent piece, how many times it has escaped by moving to another board position, how many times it has remained

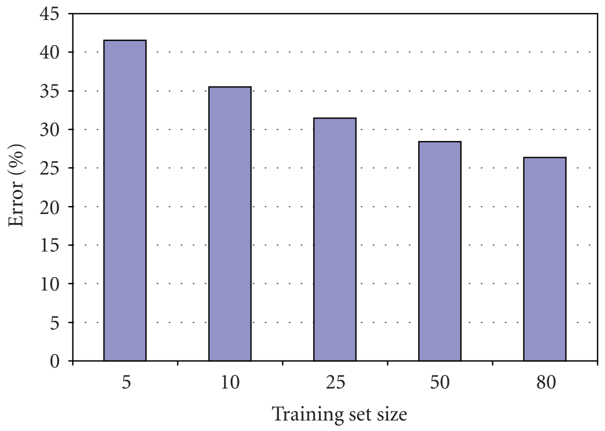

Figure 6: Averaged leave-one-out error obtained by the machine learning algorithm varying the number of game logs used for training.

on its position, and how many times it has moved from its square to threat another opponent's piece).

To build the prototype for a player, our algorithm needs first to collect data, that is, the training set, from previous game sessions with the same human player. For each of these sessions and for each piece a corresponding feature vector is built according to the criteria above which are based on the behavior of the piece in the game. The prototype for good (or bad) pieces is then determined as the average among the feature vectors representing good (or bad) pieces. More formally, given $G = \{g_1, \ldots, g_n\}$ the set of feature vectors for good pieces of a given player, and $B = \{b_1, \ldots, b_n\}$ the set of available feature vectors for bad pieces of the same player, the prototype vectors are computed as:

$$P_G = \frac{1}{n}\sum_{i=1}^{n} g_i, \qquad P_B = \frac{1}{n}\sum_{i=1}^{n} b_i. \tag{1}$$

Now, let be given a new feature vector f representing the profile of a piece of unknown type on the board; a badness score can hence be computed by using the normalized distance with respect to the player prototypes, that is,

$$s(f) = \frac{d(f, P_G) - d(f, P_B)}{d(f, P_G) + d(f, P_B)}, \tag{2}$$

where $d(x, y)$ is the Euclidean distance between vectors x and y. Note that the score is always a number between -1 (definitely good) and $+1$ (definitely bad).

On each move in the middle of a game session, the prediction of the type of the pieces on the board is performed in the following way. First, since the exact number of good and bad pieces (n_g and n_b, resp.) still on the board is a known information, a badness score for each of these pieces can be computed by utilizing (2). Then, the pieces are ranked based on this score and the n_b highest score pieces are predicted to be bad pieces.

The error committed in a prediction is computed as the number of bad pieces which are actually predicted as good ones. Needless to say, with the ranking method we used to discriminate between bad and good pieces, this error also corresponds to the number of good pieces which are incorrectly predicted as bad.

6. Experimental Results

In this section, experimental results showing the effectiveness of our profiling methodology are reported. The experimental setting consisted in a mobile phone-based version of *Ghosts.* A user played a set of 81 games against other human players. Game logs of these matches, containing the initial state of the board and all the moves of the two players, have been stored during the games.

6.1. Evaluation of the Profile Construction in a Single Game Session. First, we studied the reliability of our profiler during a game session. Clearly, at the beginning of a game, the profile cannot be precise as some features of a piece may still be unavailable or underestimated. Thus, even if the models generated by our prototype-based machine learning algorithm were perfect, the correct classification of a piece may be difficult at the early stage of a game session, given the low informative degree of its profile. We can expect the prediction error to decrease whenever the profile becomes more and more informative and complete. However, this represents a desirable property as, in general, it is not so important to have a very low error when the game is at the very beginning, whereas it becomes crucial as the game session proceeds.

In particular, plots in Figure 5 give the percentage of error that affects our machine learning algorithm during ten single game sessions. Instantaneous values of the error percentage are provided at each move performed during the game session. Needless to say, the total number of moves to conclude a game session is variable; this is why charts in Figure 5 have different ranges of the values on the *x*-axis. Note that 50% of error means that two good (and two bad) ghosts out of four were wrongly labeled as bad (good), 25% represents the case with only one good (and one bad) ghost wrongly labeled, and 0% error is the case when the system provides a perfect prediction of the type of all the ghosts in the board. As anticipated, the precision of the predictor quickly improves as a game session continues and six out of the ten charts in Figure 5 achieve a percentage of 0% error during the second half of the game session.

6.2. Evaluation of the Machine Learning Algorithm. In a second set of experiments, we evaluated the performance of the machine learning algorithm. This has been done by estimating the probability to make errors in future match games. To this end, a leave-one-out procedure has been used. This measure, very common in the machine learning community for evaluating the expected performance of a classifier, provably gives a statistically unbiased estimate of the expected percentage of mistakes that a classifier will make. To compute these statistics, for each available game session log, a model has been generated by removing that game log from the set and training the prototype-based classifier on the remaining 80 game logs. Then, the mean error in the left out game log is computed as the proportion of piece type guess mistakes out of the total guesses during the game. Finally, all these mean errors have been collected and averaged to obtain a final estimate of the algorithm's performance. In particular, with the considered set of 81 game logs, the leave-one-out estimate resulted in a 26.3% error.

We also wanted to show how the prototype-based classifier improves its accuracy as the number of previously stored game logs increases. To this aim, we have exploited again the leave-one-out procedure but this time considering different training set sizes (consisting of 5, 10, 25, 50, 80 game logs). For each chosen training log set sizes, the leave-one-out procedure has been repeated each time considering a different left out game log; the averaged resulting error percentages are reported in Figure 6. As expected, with a larger size of the training set, the machine learning algorithm is able to build a more reliable prototype of the player's profile, thus improving the performance of the prediction system.

7. Conclusion and Future Work

The mass adoption of mobile phones in our society has transformed them from gadgets into commodities. The technical features of these devices have reached a quality level that makes them able to run multimedia applications. One of the most successful mobile applications is certainly represented by gaming and, in this paper, we discussed technical issues related to this context. In particular, we have proposed an original solution that improves the capability of the AI by allowing it to profile its human opponent and exploit her/his weaknesses. Our approach is particularly useful for imperfect information games and for games running on devices with limited computational capabilities (e.g., mobile phones), where a pure searching algorithm could not be employed; at the same time, it can be easily plugged into any standard AI or temporal difference learning-based algorithm to enhance its performance.

As a real testbed for our solution, we created a mobile phone-based game also considering the state-of-the-art in compatibility and connectivity technology available on today's phones. The game that we have chosen for our experiments is *Ghosts,* an imperfect information game that can be either played against another player or against the AI algorithm. Results gathered during our experimental evaluation demonstrate that our approach achieves the desired goal of an effective profiling of the player.

Intentionally, we have chosen a very simple machine learning technique for our experiments for two main reasons. First, we wanted to be sure that the chosen algorithm could be run, producing the desired goal, even on a device with limited computational capabilities. Second, our experiments were intended to prove the viability of our general method and they achieved this goal; in the future, more complex machine learning algorithms such as, for instance, SVM [28] can be utilized; also, an extended set of features to profile the opponent's behavior could be evaluated.

Another attractive future direction for this work would be that of applying recent machine learning methods which can deal with problems different from the classification one. In fact, it can be noted that the reduction of the *Ghosts* game to a classification problem is not the only one possible. Actually, the real need in the *Ghosts* game is to decide which the good and bad pieces are once we have a rank of the pieces based on their "badness;" then, ad hoc approaches to learn rankings and preferences (e.g., [29]) can be exploited to improve the performance of the system.

Finally, we also plan to apply our solution to more complex imperfect information games such as *Invisible Chess* and *Kriegspiel* which are heterodox chess variation in which players are not informed of their opponents position and moves [30, 31].

Acknowledgments

The authors feel indebted toward the anonymous referees, whose insightful comments and suggestions greatly helped them in improving the final version of this manuscript. Furthermore, their deep gratitude goes to Federico Giardina and Ivan Mazzarelli for their technical contribution in developing the digital version of the game and the experimental setup.

References

[1] "Homo Mobilis," The Economist, 2008, http://www.economist.com/specialreports/displaystory.cfm?story_id=10950487/.

[2] "Cellular Mobile Gaming Grows to EUR 6 Billion in 2006," 2003, http://www.cellular.co.za/news_2003/101503-mobile_gaming_grows_to_eur_6_bil.htm.

[3] B. Gibson, "Casual Gamers and Female Gamers to Drive Mobile Games Revenues over the $10 Billion Mark by 2009," http://www.juniperresearch.com/shop/viewpressrelease.php?id=19&pr=16.

[4] ZDNet Personal Tech, "Mobile-game market to boom," 2004, http://news.zdnet.com/2100-1040_22-5355648.html.

[5] J. O. B. Soh and B. C. Y. Tan, "Mobile gaming," *Communications of the ACM*, vol. 51, no. 3, pp. 35–39, 2008.

[6] E. Koivisto, "Mobile games 2010," in *Proceedings of the International Conference on Game Research and Development (CyberGames '06)*, pp. 1–2, Murdoch University, Perth, Australia, December 2006.

[7] Symbian OS: the Open Mobile Operating System, http://www.symbian.com/.

[8] Qualcomm Brew, http://brew.qualcomm.com/brew/en/.

[9] "The Java ME Platform—the Most Ubiquitous Application Platform for Mobile Devices," http://java.sun.com/javame/index.jsp.

[10] Bluetooth.com | The Official Bluetooth® Technology Info Site, http://www.bluetooth.com/bluetooth.

[11] IEEE 802.11-2007, "IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2007.

[12] E. Gustafsson and A. Jonsson, "Always best connected," *IEEE Wireless Communications*, vol. 10, no. 1, pp. 49–55, 2003.

[13] V. Gazis, N. Houssos, N. Alonistioti, and L. Merakos, "On the complexity of "always best connected" in 4G mobile networks," in *Proceedings of the 58th IEEE Vehicular Technology Conference (VTC '03)*, vol. 4, pp. 2312–2316, Orlando, Fla, USA, October 2003.

[14] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 211–229, 1959.

[15] M. Campbell, "Knowledge discovery in deep blue," *Communications of the ACM*, vol. 42, no. 11, pp. 65–67, 1999.

[16] V. Allis, *A knowledge-based approach of connect-four the game is solved: white wins*, M.S. thesis, Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands, 1998.

[17] R. Gasser, *Efficiently harnessing computational resources for exhaustive search*, Ph.D. thesis, Eidgenössische Technische Hochschule, Zurich, Switzerland, 1995.

[18] M. Buro, "The Othello match of the year: Takeshi Murakami vs. Logistello," *International Computer Chess Association Journal*, vol. 20, no. 3, pp. 189–193, 1997.

[19] D. Billings, "The first international RoShamBo programming competition," *International Computer Games Association Journal*, vol. 23, no. 1, pp. 42–50, 2000.

[20] D. Egnor, "Iocaine powder," *International Computer Games Association Journal*, vol. 23, no. 1, pp. 33–35, 2000.

[21] T. Mitchell, *Machine Learning*, McGraw Hill, New York, NY, USA, 1997.

[22] J. Schaeffer, "The games computers (and people) play," *Advances in Computers*, vol. 50, pp. 189–266, 2000.

[23] D. Billings, L. Pena, J. Schaeffer, and D. Szafron, "Using probabilistic knowledge and simulation to play poker," in *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference (AAAI '99)*, pp. 697–703, Orlando, Fla, USA, July 1999.

[24] M. L. Ginsberg, "GIB: steps toward an expert-level bridge-playing program," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*, pp. 584–589, Stockholm, Sweden, July-August 1999.

[25] B. Sheppard, "Mastering Scrabble," *IEEE Intelligent Systems*, vol. 14, no. 6, pp. 15–16, 1999.

[26] F. Aiolli and C. E. Palazzi, "Enhancing artificial intelligence in games by learning the opponent's playing style," in *Proceedings of the 1st IFIP Entertainment Computing Symposium on New Frontiers for Entertainment Computing, in Conjunction with the 20th IFIP World Computer Congress (ECS '08)*, pp. 1–10, Milan, Italy, September 2008.

[27] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall Series in Artificial Intelligence, Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

[28] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, Berlin, Germany, 1995.

[29] F. Aiolli, "A preference model for structured supervised learning tasks," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05)*, pp. 557–560, Houston, Tex, USA, November 2005.

[30] A. Bud, D. Albrecht, A. Nicholson, and I. Zukerman, "Playing invisible chess with information-theoretic advisors," in *Proceedings of the AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*, Stanford, Calif, USA, March 2001.

[31] P. Ciancarini, F. Dalla Libera, and F. Maran, "Decision making under uncertainty: a rational approach to Kriegspiel," *Advances in Computer Chess*, vol. 8, pp. 277–298, 1997.