US 20090093287A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2009/0093287 A1

(43) Pub. Date: Apr. 9, 2009

Herbrich et al.

(54) **DETERMINING RELATIVE PLAYER SKILLS AND DRAW MARGINS**

(75) Inventors: **Ralf Herbrich**, Cambridge (GB); **Thore Graepel**, Cambridge (GB); **Thomas Minka**, Cambridge (GB); **Pierre Dangauthier**, Grenoble (FR)

Correspondence Address:
LEE & HAYES, PLLC
601 W. RIVERSIDE AVENUE, SUITE 1400
SPOKANE, WA 99201 (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(21) Appl. No.: **11/869,165**

(22) Filed: **Oct. 9, 2007**

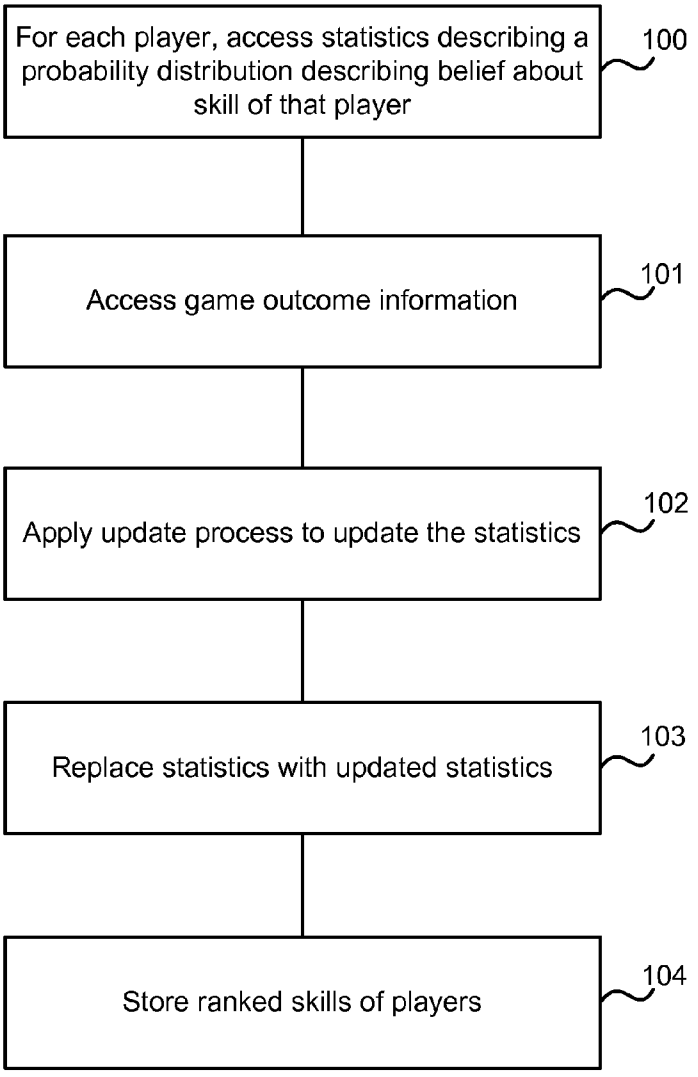
Publication Classification

(51) **Int. Cl.**
A63F 9/24 (2006.01)

(52) **U.S. Cl.** **463/1**

(57) **ABSTRACT**

A process for determining relative player skills and draw margins is described. Information about an outcome of a game between at least a first player opposing a second player is received. Also, for each player, skill statistics are received associated with a distribution representing belief about skill of that player. Draw margin statistics are received associated with a distribution representing belief about ability of that player to force a draw. An update process is performed to update the statistics on the basis of the received information about the game outcome. In an embodiment a Bayesian inference process is used during the update process which may take past and future player achievement into account.



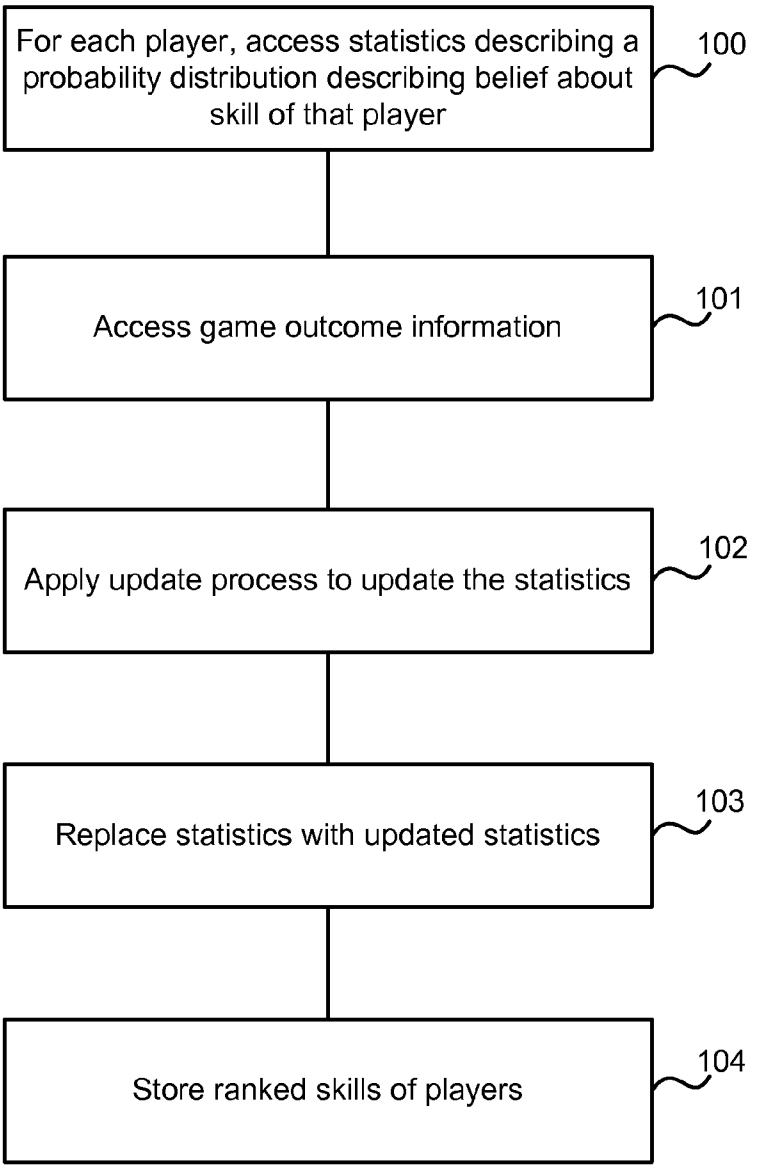

For each player, access statistics describing a probability distribution describing belief about skill of that player
100
Access game outcome information
101
Apply update process to update the statistics
102
Replace statistics with updated statistics
103
Store ranked skills of players
104

FIG. 1

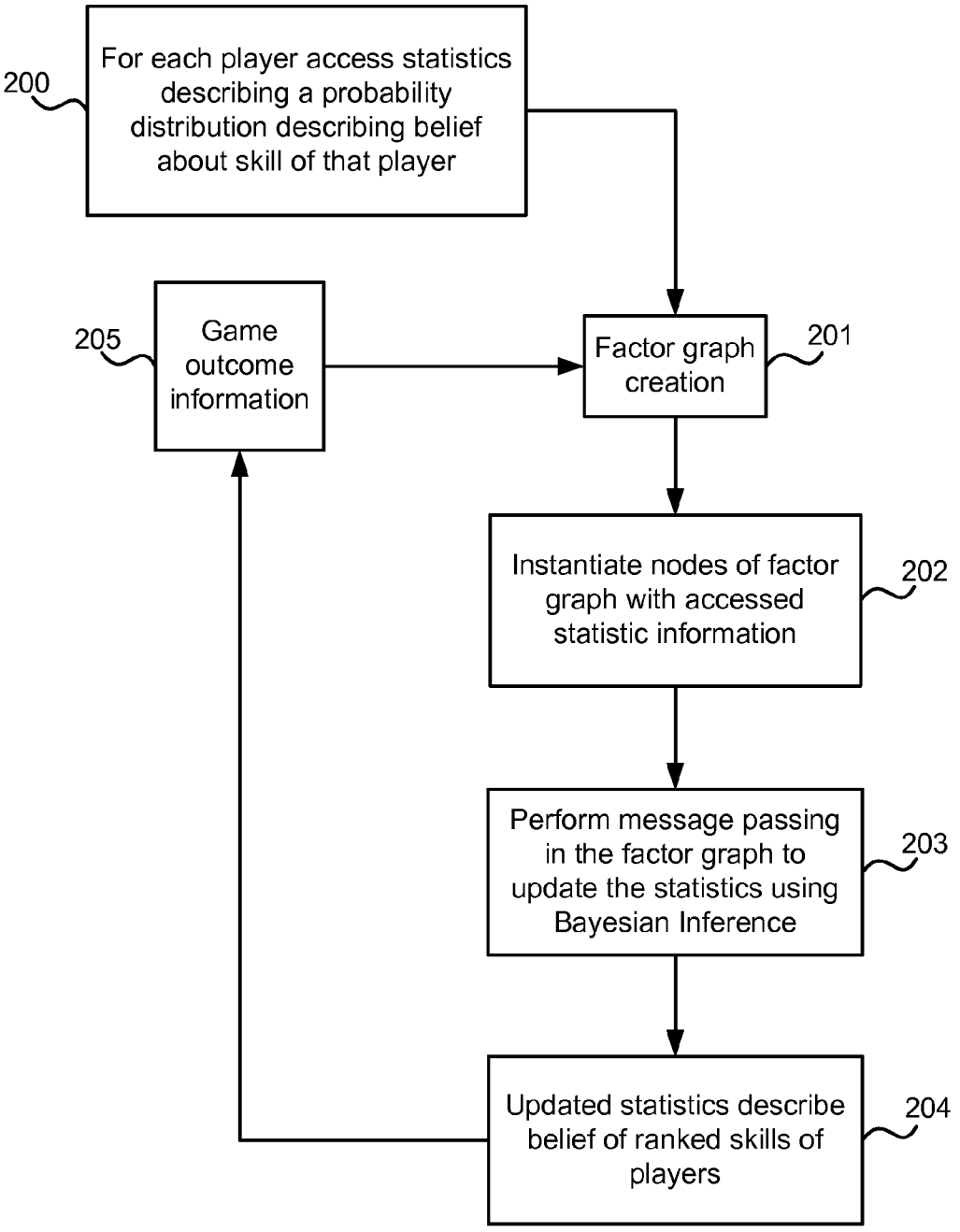

200
For each player access statistics describing a probability distribution describing belief about skill of that player
205
Game outcome information
Factor graph creation
201
Instantiate nodes of factor graph with accessed statistic information
202
Perform message passing in the factor graph to update the statistics using Bayesian Inference
203
Updated statistics describe belief of ranked skills of players
204

FIG. 2

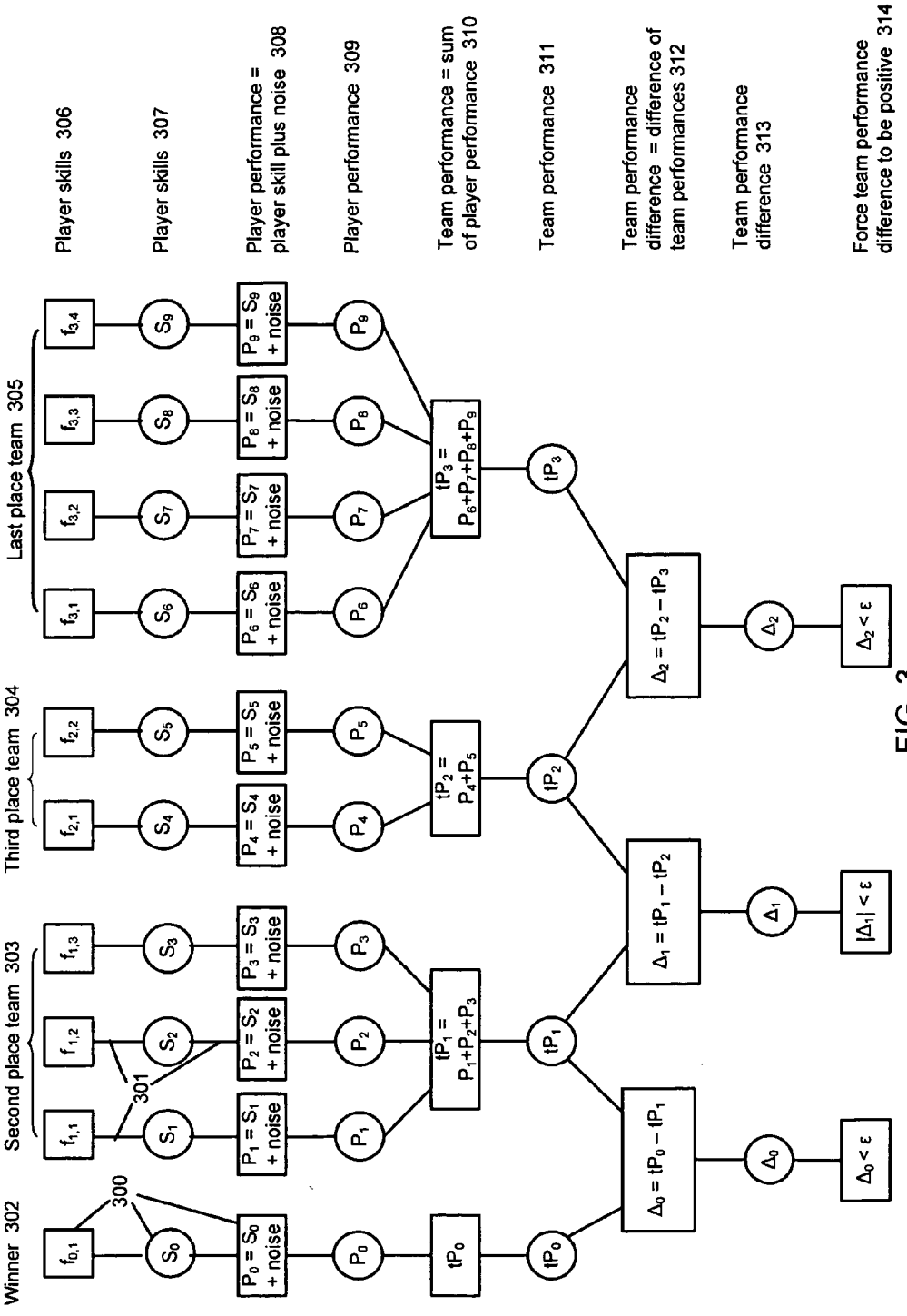

Winner 302
Second place team 303
Third place team 304
Last place team 305
Player skills 306
Player skills 307
Player performance = player skill plus noise 308
Player performance 309
Team performance = sum of player performance 310
Team performance 311
Team performance difference = difference of team performances 312
Team performance difference 313
Force team performance difference to be positive 314
300
301
$f_{0,1}$
$f_{1,1}$
$f_{1,2}$
$f_{1,3}$
$f_{2,1}$
$f_{2,2}$
$f_{3,1}$
$f_{3,2}$
$f_{3,3}$
$f_{3,4}$
S_0
S_1
S_2
S_3
S_4
S_5
S_6
S_7
S_8
S_9
$P_0 = S_0 + \text{noise}$
$P_1 = S_1 + \text{noise}$
$P_2 = S_2 + \text{noise}$
$P_3 = S_3 + \text{noise}$
$P_4 = S_4 + \text{noise}$
$P_5 = S_5 + \text{noise}$
$P_6 = S_6 + \text{noise}$
$P_7 = S_7 + \text{noise}$
$P_8 = S_8 + \text{noise}$
$P_9 = S_9 + \text{noise}$
P_0
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
tP_0
$tP_1 = P_1+P_2+P_3$
$tP_2 = P_4+P_5$
$tP_3 = P_6+P_7+P_8+P_9$
tP_0
tP_1
tP_2
tP_3
$\Delta_0 = tP_0 - tP_1$
$\Delta_1 = tP_1 - tP_2$
$\Delta_2 = tP_2 - tP_3$
Δ_0
Δ_1
Δ_2
$\Delta_0 < \varepsilon$
$|\Delta_1| < \varepsilon$
$\Delta_2 < \varepsilon$

FIG. 3

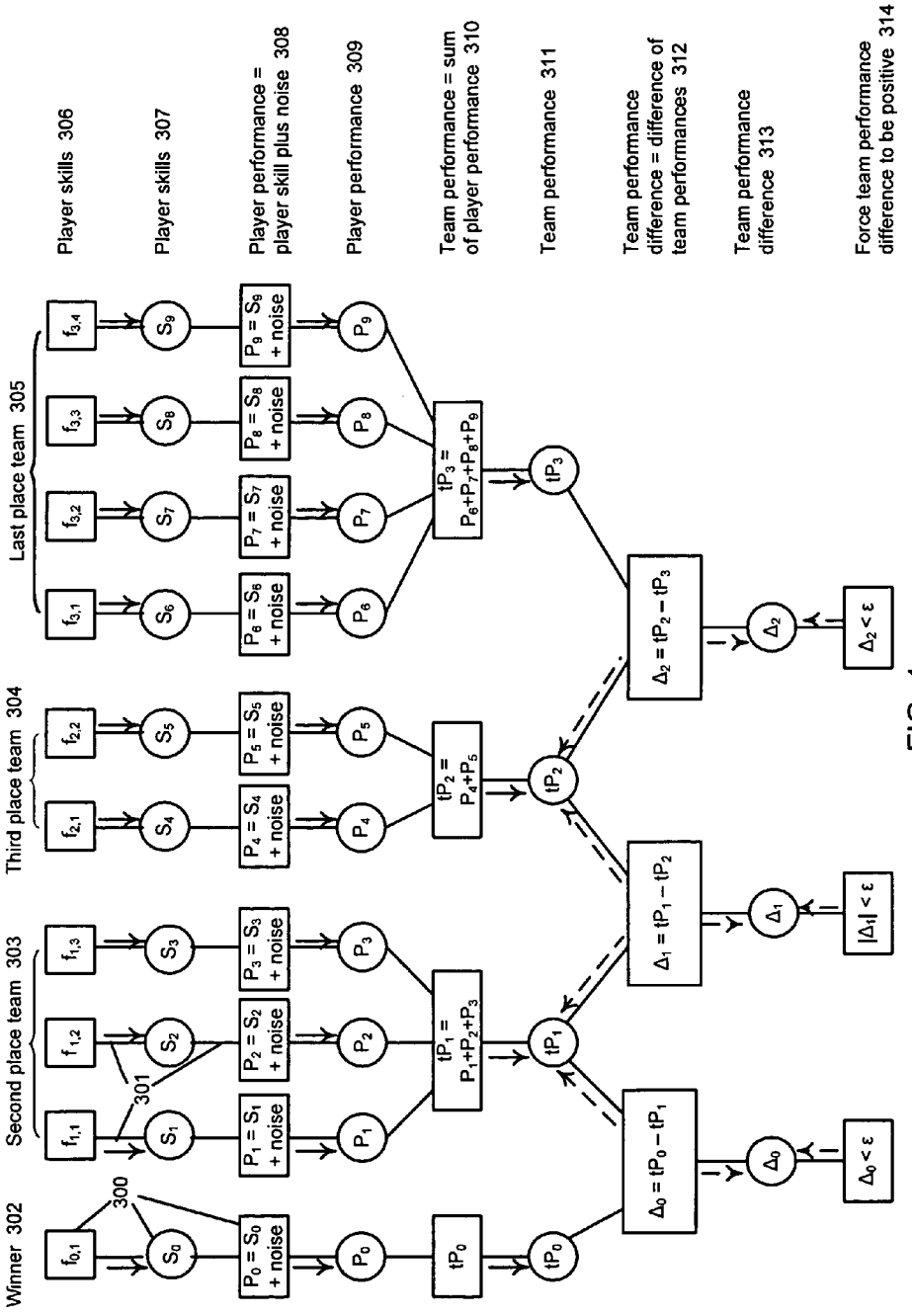
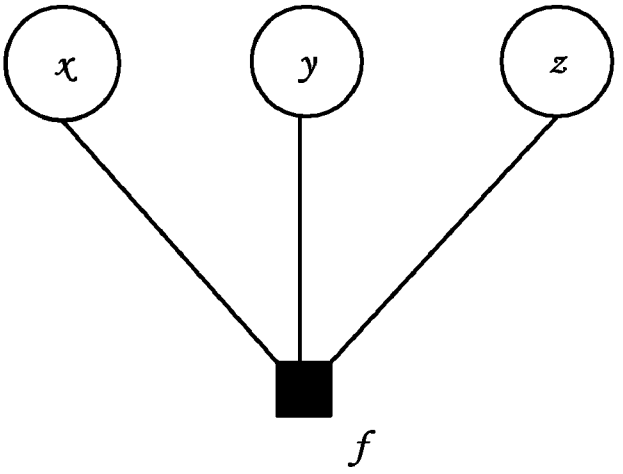

Winner 302
Second place team 303
Third place team 304
Last place team 305
Player skills 306
Player skills 307
Player performance = player skill plus noise 308
Player performance 309
Team performance = sum of player performance 310
Team performance 311
Team performance difference = difference of team performances 312
Team performance difference 313
Force team performance difference to be positive 314
300
301
$f_{0,1}$
$f_{1,1}$
$f_{1,2}$
$f_{1,3}$
$f_{2,1}$
$f_{2,2}$
$f_{3,1}$
$f_{3,2}$
$f_{3,3}$
$f_{3,4}$
S_0
S_1
S_2
S_3
S_4
S_5
S_6
S_7
S_8
S_9
$P_0 = S_0 + \text{noise}$
$P_1 = S_1 + \text{noise}$
$P_2 = S_2 + \text{noise}$
$P_3 = S_3 + \text{noise}$
$P_4 = S_4 + \text{noise}$
$P_5 = S_5 + \text{noise}$
$P_6 = S_6 + \text{noise}$
$P_7 = S_7 + \text{noise}$
$P_8 = S_8 + \text{noise}$
$P_9 = S_9 + \text{noise}$
P_0
P_1
P_2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
tP_0
$tP_1 = P_1+P_2+P_3$
$tP_2 = P_4+P_5$
$tP_3 = P_6+P_7+P_8+P_9$
tP_0
tP_1
tP_2
tP_3
$\Delta_0 = tP_0 - tP_1$
$\Delta_1 = tP_1 - tP_2$
$\Delta_2 = tP_2 - tP_3$
Δ_0
Δ_1
Δ_2
$\Delta_0 < \varepsilon$
$|\Delta_1| < \varepsilon$
$\Delta_2 < \varepsilon$

FIG. 4



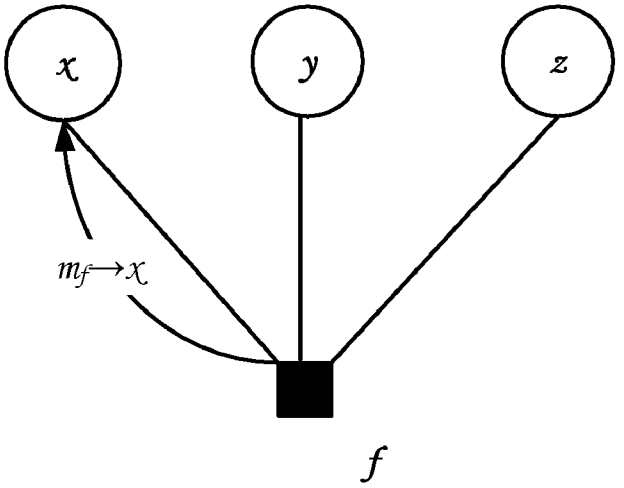


FIG. 5

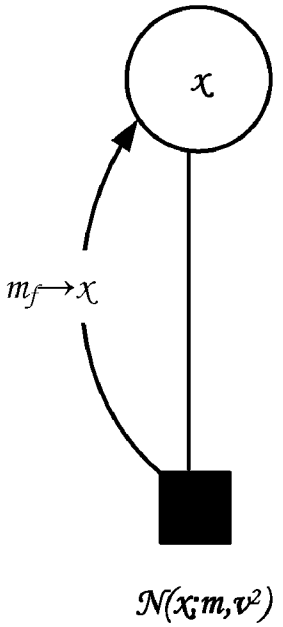


FIG. 6

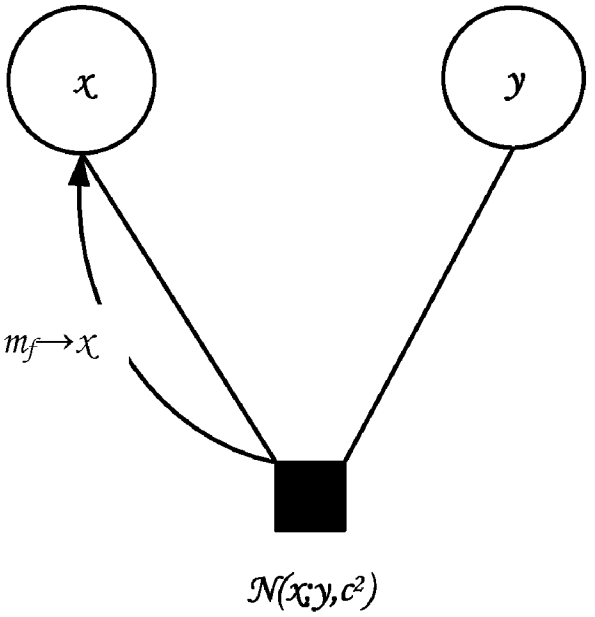
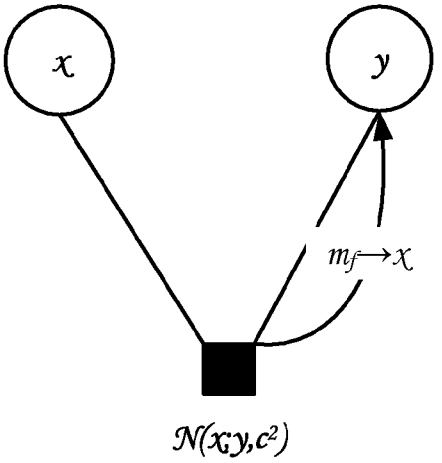


FIG. 7


x
y
$m_f \rightarrow x$
$\mathcal{N}(x;y,c^2)$

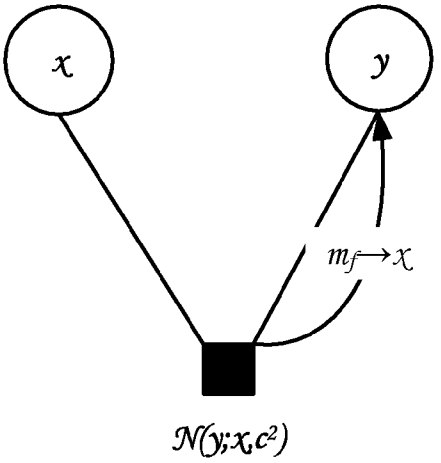

x
y
$m_f \rightarrow x$
$\mathcal{N}(y;x,c^2)$

FIG. 8

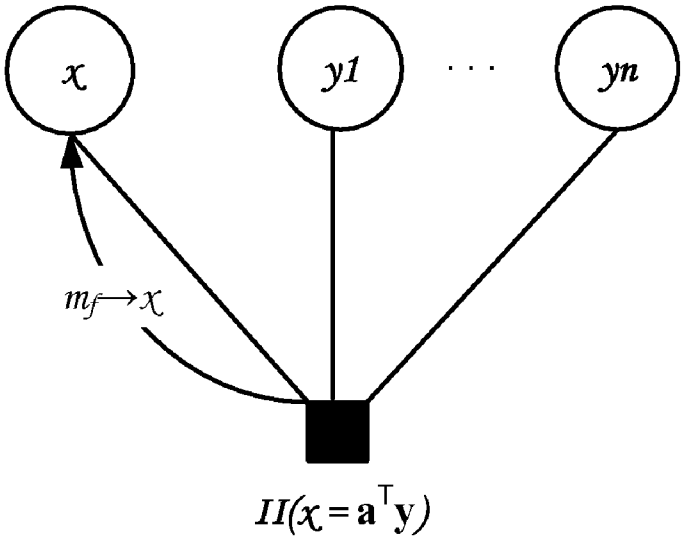


FIG. 9

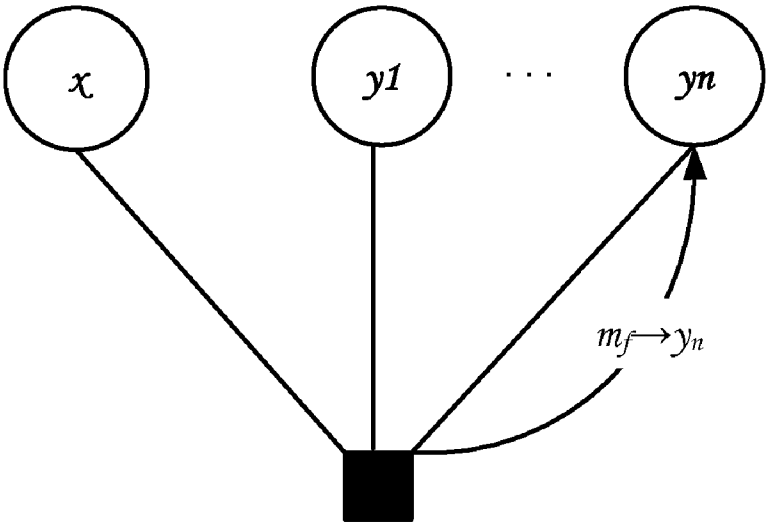
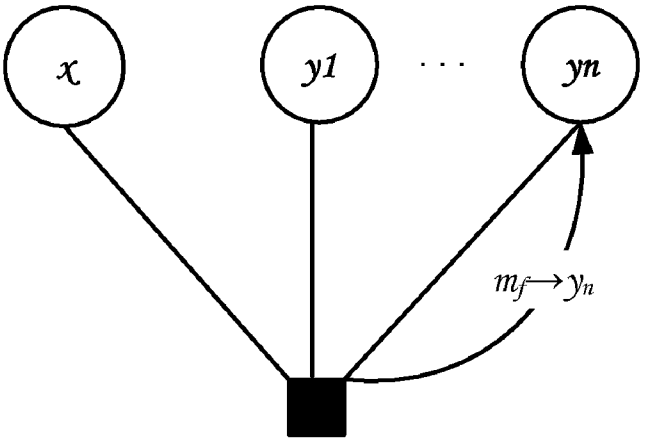


FIG. 10



$$\mathbf{a} = \frac{1}{b_n} \cdot \begin{bmatrix} -b_1 \\ \vdots \\ -b_n - 1 \\ +1 \end{bmatrix}$$

$$II(y_n = \mathbf{a}^{\top}\mathbf{y}[y_1, \ldots, y_n\text{-}1, x])$$

FIG. 11

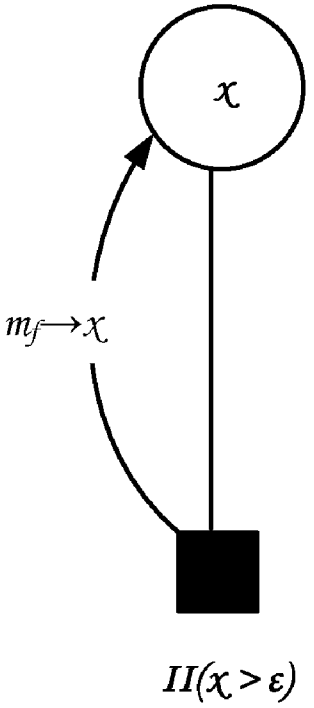
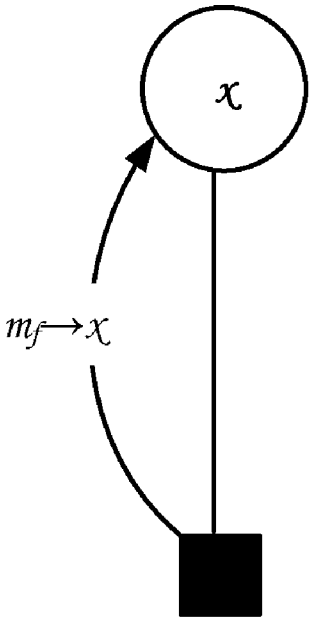


FIG. 12



$II(|x| \leq \varepsilon)$

FIG. 13

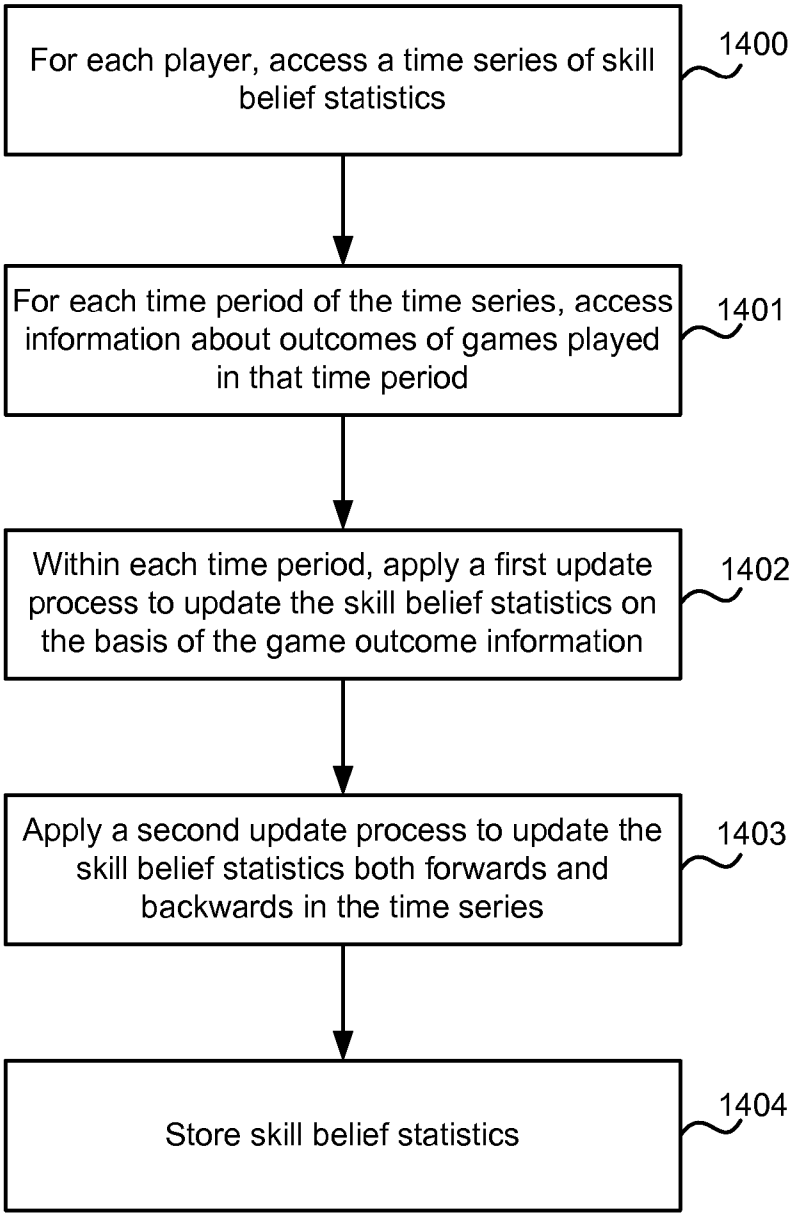

For each player, access a time series of skill belief statistics
1400
For each time period of the time series, access information about outcomes of games played in that time period
1401
Within each time period, apply a first update process to update the skill belief statistics on the basis of the game outcome information
1402
Apply a second update process to update the skill belief statistics both forwards and backwards in the time series
1403
Store skill belief statistics
1404

FIG. 14

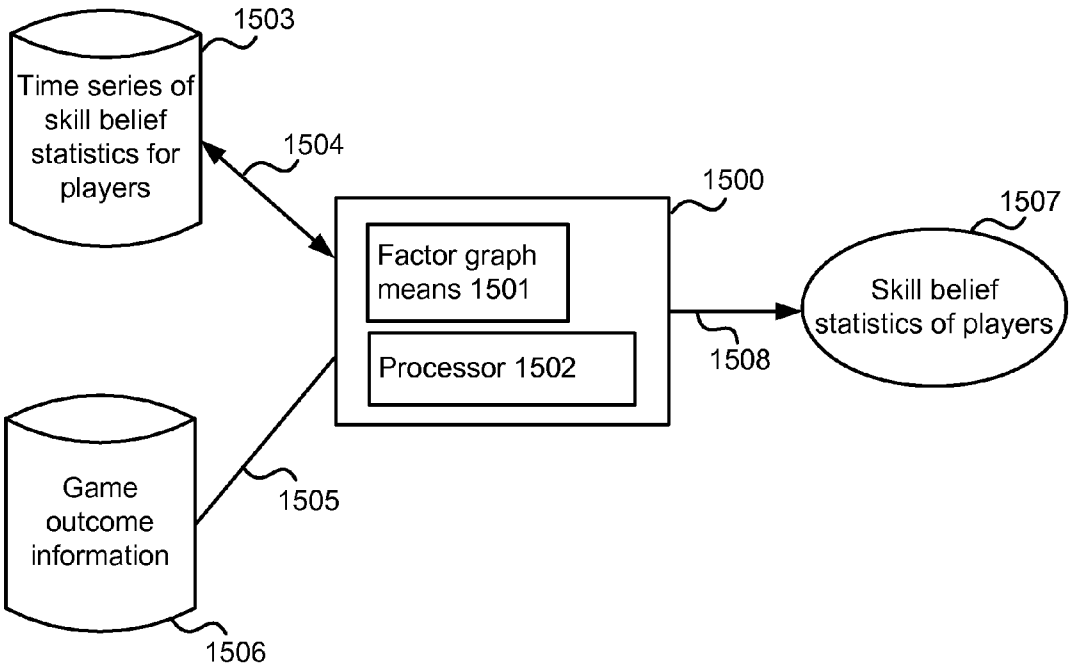

1503
Time series of skill belief statistics for players
1504
1500
1507
Factor graph means 1501
Processor 1502
Skill belief statistics of players
1508
Game outcome information
1505
1506

FIG. 15

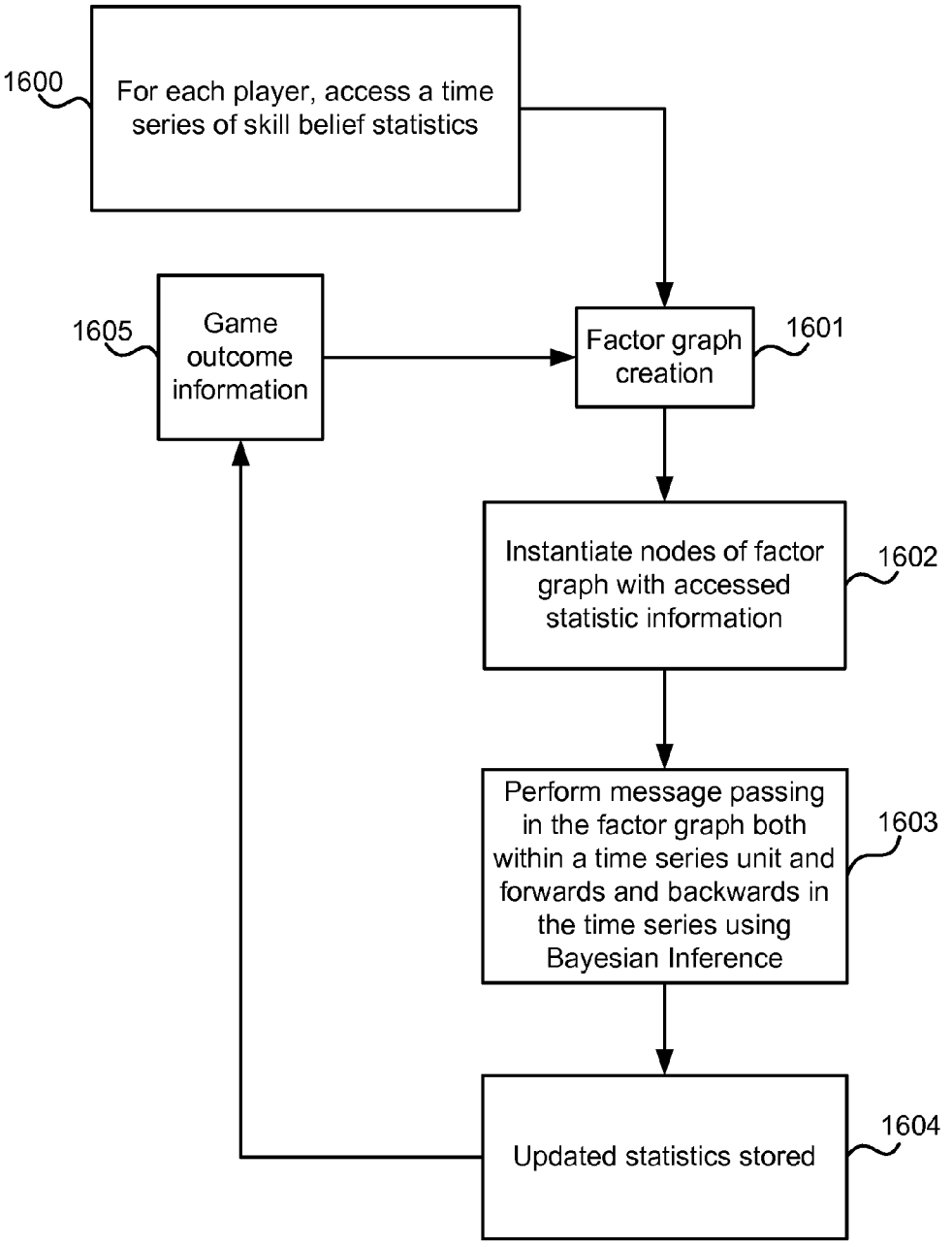

1600
For each player, access a time series of skill belief statistics
1605
Game outcome information
Factor graph creation
1601
Instantiate nodes of factor graph with accessed statistic information
1602
Perform message passing in the factor graph both within a time series unit and forwards and backwards in the time series using Bayesian Inference
1603
Updated statistics stored
1604

FIG. 16

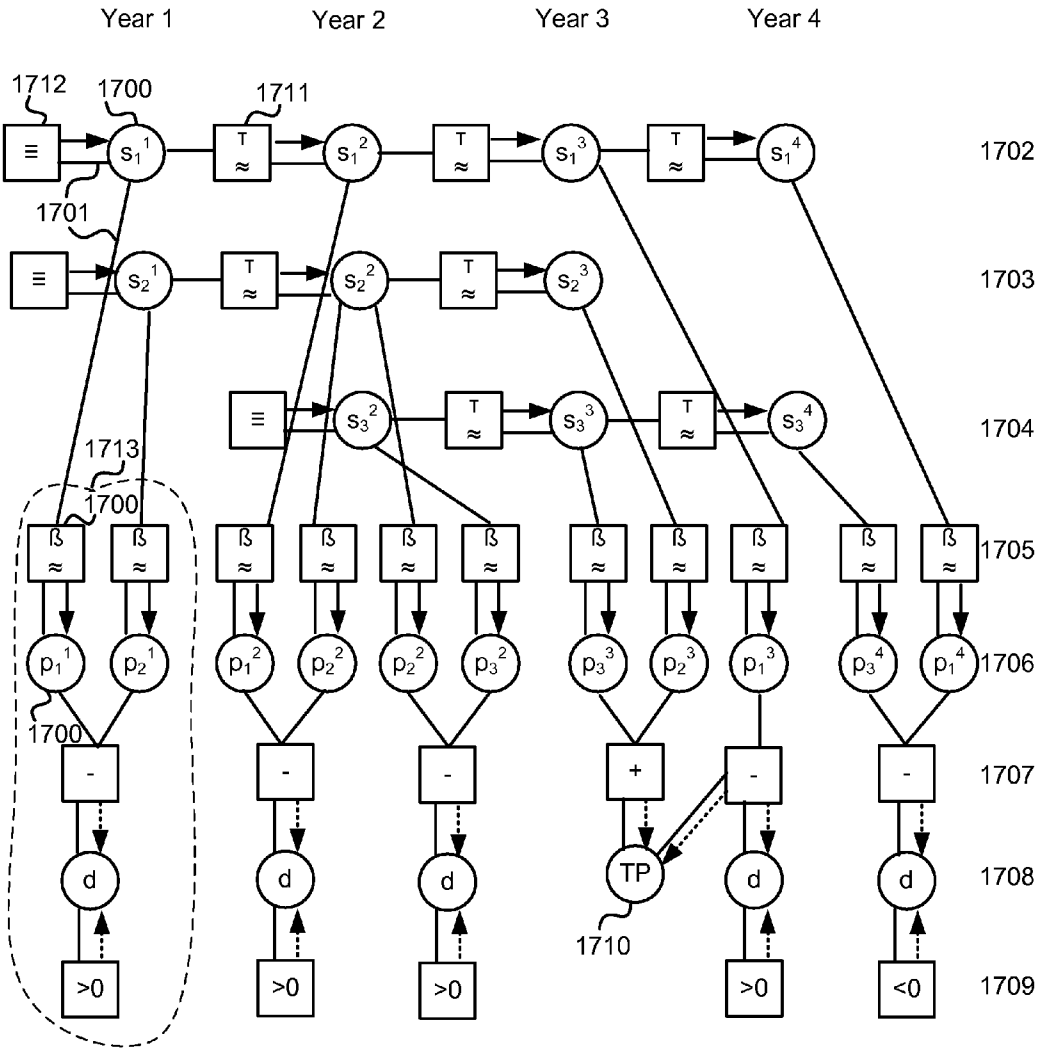

Year 1
Year 2
Year 3
Year 4
1712
1700
1711
1701
1713
1700
1700
1710
1702
1703
1704
1705
1706
1707
1708
1709
TP
>0
<0

FIG. 17

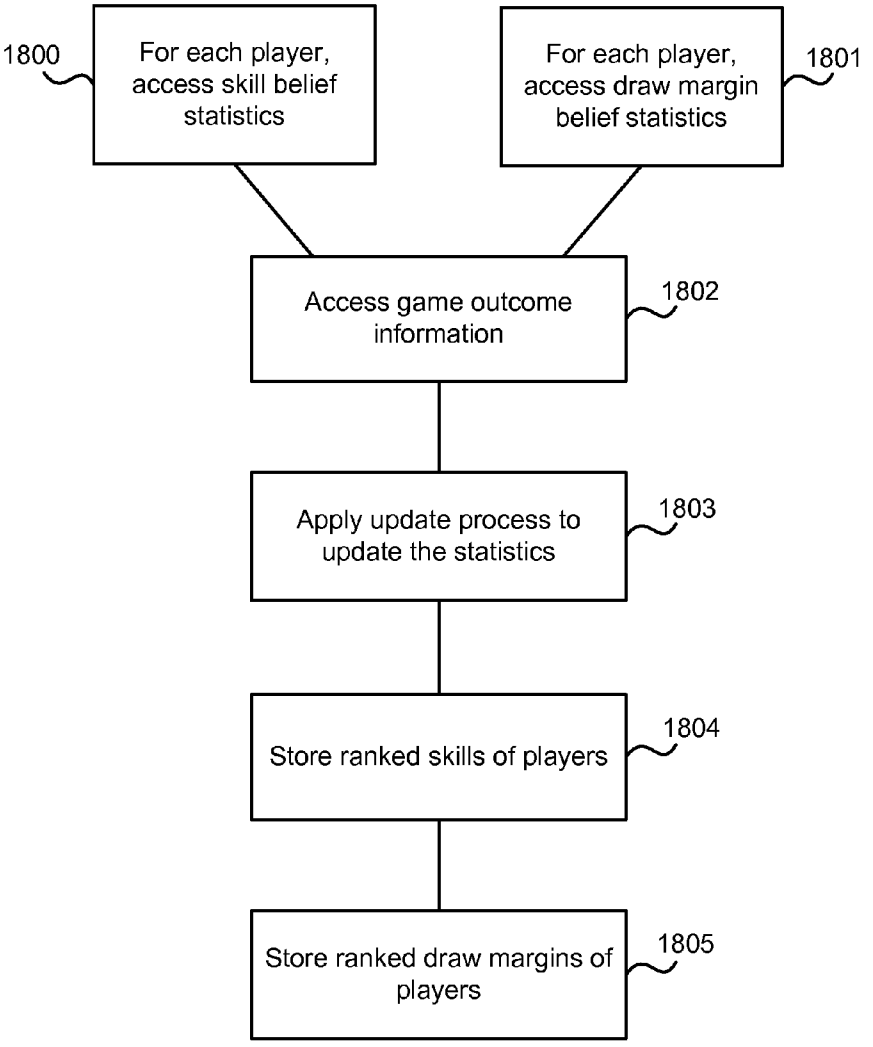

1800
For each player, access skill belief statistics
For each player, access draw margin belief statistics
1801
Access game outcome information
1802
Apply update process to update the statistics
1803
Store ranked skills of players
1804
Store ranked draw margins of players
1805

FIG. 18

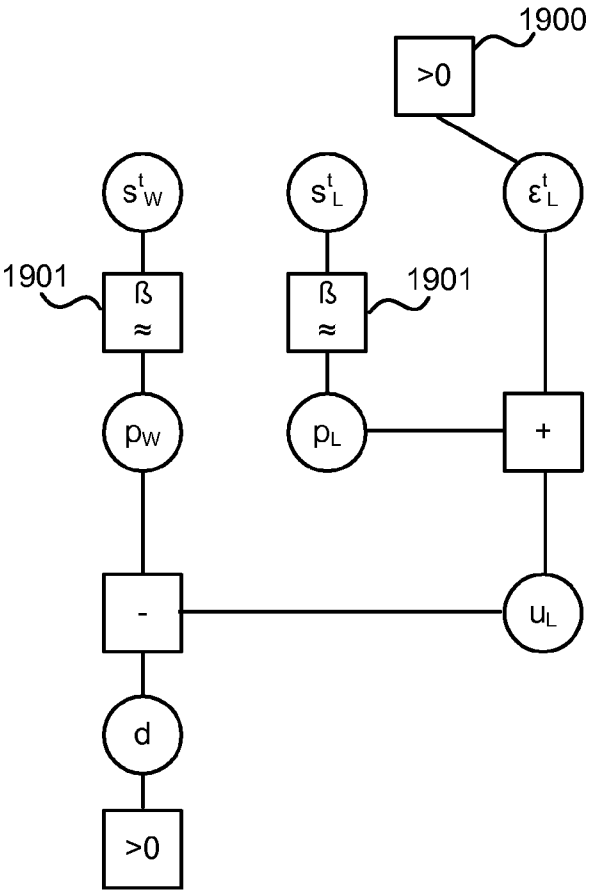


FIG. 19

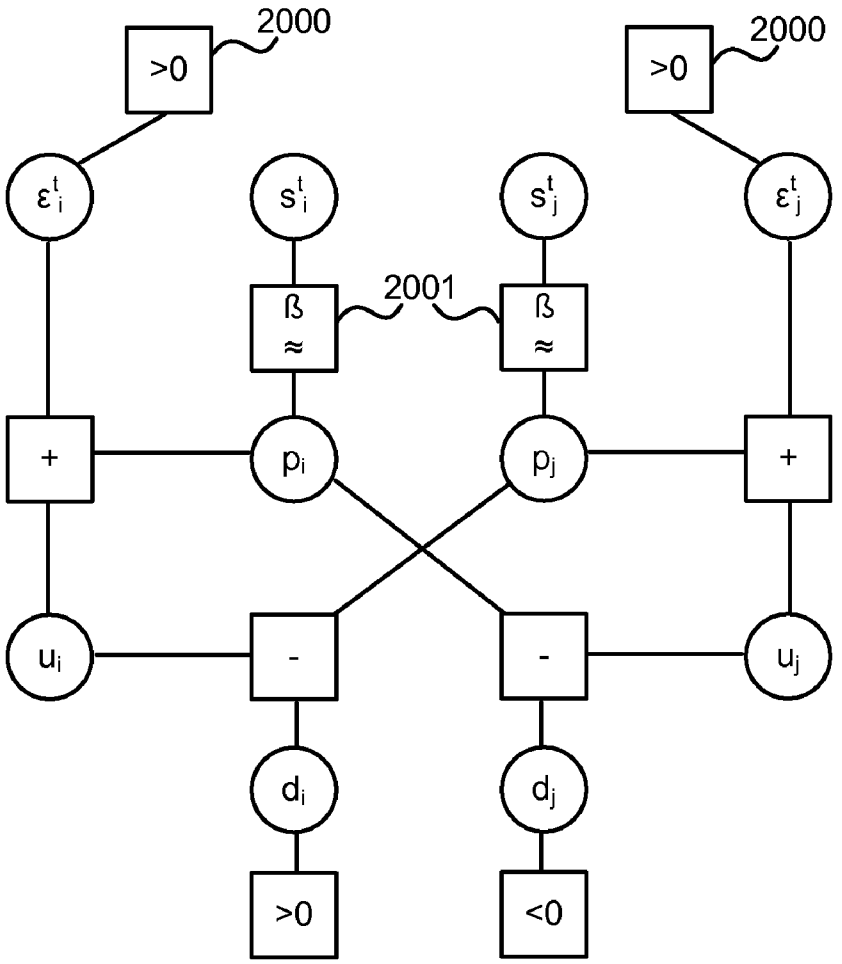

2000
>0
2000
>0
ε_i^t
s_i^t
s_j^t
ε_j^t
2001
ß
≈
ß
≈
+
p_i
p_j
+
u_i
-
-
u_j
d_i
d_j
>0
<0

FIG. 20

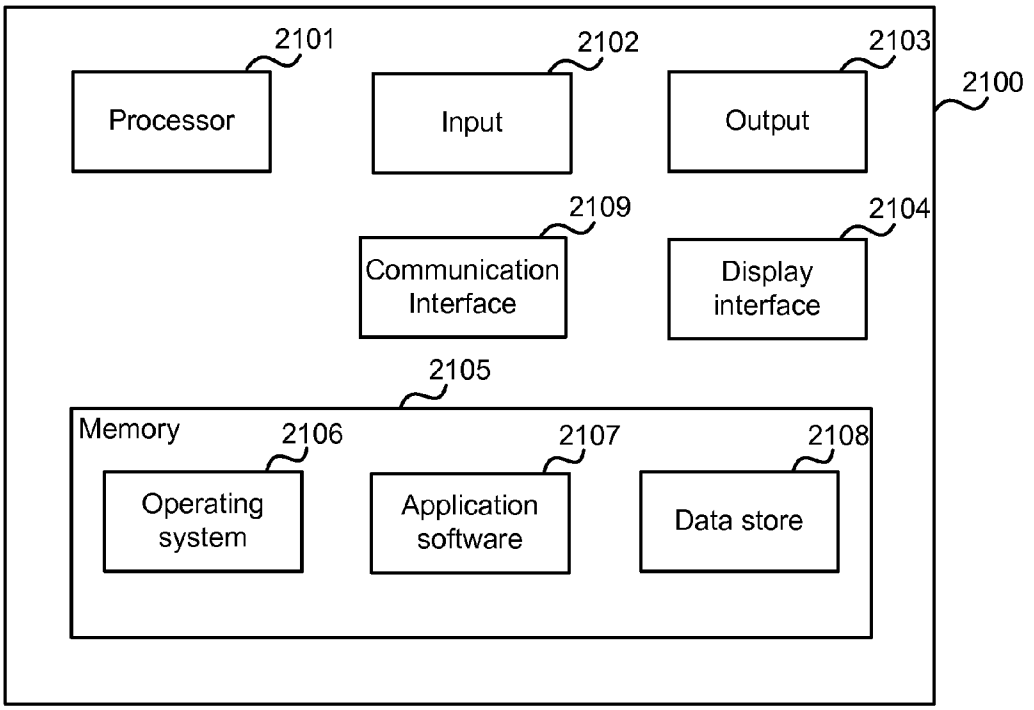

2101
Processor
2102
Input
2103
Output
2100
2109
Communication Interface
2104
Display interface
2105
Memory
2106
Operating system
2107
Application software
2108
Data store

FIG. 21

DETERMINING RELATIVE PLAYER SKILLS AND DRAW MARGINS

BACKGROUND

[0001] There is a desire to provide a way to determine relative skills of players of games such as computer games, chess, tennis, and any other type of game. This needs to be achieved in a manner whereby the indication of relative skill is as accurate as possible and also is understood and accepted by end users (i.e. game players). In addition, the relative skills need to be determined quickly even in the case of games involving many players and also in the case of many teams of players, each team having many members. This is particularly problematic because in these situations, computation complexity typically increases significantly. That is, there is a need to determine skills of players after very few game outcomes and/or with computational efficiency. Players can be human players or computer programs.

[0002] Bayesian statistical techniques have been used to determine indications of player relative skill. However, it is desired to improve the accuracy of these previous approaches whilst at the same time addressing the issues of computational complexity.

[0003] The embodiments of the invention described below are not limited to implementations which solve any or all of the disadvantages mentioned above.

SUMMARY

[0004] The following presents a simplified summary of the disclosure in order to provide a basic understanding to the reader. This summary is not an extensive overview of the disclosure and it does not identify key/critical elements of the invention or delineate the scope of the invention. Its sole purpose is to present some concepts disclosed herein in a simplified form as a prelude to the more detailed description that is presented later.

[0005] A process for determining relative player skills and draw margins is described. Information about an outcome of a game between at least a first player opposing a second player is received. Also, for each player, skill statistics are received associated with a distribution representing belief about skill of that player. Draw margin statistics are received associated with a distribution representing belief about ability of that player to force a draw. An update process is performed to update the statistics on the basis of the received information about the game outcome. In an embodiment a Bayesian inference process is used during the update process which may take past and future player achievements into account.

[0006] Many of the attendant features will be more readily appreciated as the same becomes better understood by reference to the following detailed description considered in connection with the accompanying drawings.

DESCRIPTION OF THE DRAWINGS

[0007] The present description will be better understood from the following detailed description read in light of the accompanying drawings, wherein:

[0008] FIG. **1** is a block diagram of an example method of ranking skills of players;

[0009] FIG. **2** is a block diagram of an example method of ranking skills of players using a factor graph;

[0010] FIG. **3** is a schematic diagram of an example factor graph;

[0011] FIG. **4** is a schematic diagram of another example factor graph;

[0012] FIGS. **5** to **13** are example factor graphs;

[0013] FIG. **14** is a block diagram of an example method of updating a time series of skill belief statistics;

[0014] FIG. **15** is a schematic diagram of a system for determining skill belief statistics of players;

[0015] FIG. **16** is a block diagram of an example method of updating a time series of skill belief statistics using message passing;

[0016] FIG. **17** is an example factor graph;

[0017] FIG. **18** is a block diagram of an example method of updating player skill belief statistics and/or player draw margin belief statistics;

[0018] FIG. **19** is an example factor graph;

[0019] FIG. **20** is another example factor graph;

[0020] FIG. **21** illustrates an exemplary computing-based device in which embodiments of a system for estimating player skill statistics and/or player draw margins may be implemented.

[0021] Like reference numerals are used to designate like parts in the accompanying drawings.

DETAILED DESCRIPTION

[0022] The detailed description provided below in connection with the appended drawings is intended as a description of the present examples and is not intended to represent the only forms in which the present example may be constructed or utilized. The description sets forth the functions of the example and the sequence of steps for constructing and operating the example. However, the same or equivalent functions and sequences may be accomplished by different examples.

[0023] As mentioned above, Bayesian statistical techniques have previously been used to determine indications of player relative skill. For example, as described in U.S. Pat. No. 7,050,868 entitled Bayesian Scoring and European patent application number EP06270014 which are both incorporated herein by reference in their entirety.

[0024] An example method for determining indications of player relative skill as described in U.S. Pat. No. 7,050,868 and EP06270014 is now summarized to aid in understanding embodiments of the present invention. At a high level, this method involves Bayesian inference techniques. Belief about a skill of each player of a game is modeled using a probability distribution of any suitable type which is described by statistics. For example, a Gaussian distribution is used which is uniquely described by its mean and standard deviation. The mean is used to model the average player skill belief and the standard deviation models the uncertainty associated with assessment of the player's skill.

[0025] As shown in FIG. **1**, for each player, statistics are accessed describing a probability distribution describing belief about skill of that player (block **100**). For example, these statistics may be set at default values for new players or may be accessed from a database of values previously found for observed players. Game outcome information is accessed (block **101**). For example, this may comprise information about winners and losers of the game, whether a draw occurred, any place ranking information and any information about whether teams of players were involved. The game may be a two player match or may involve multiple teams of players, each team having one or more players.

[0026] The system assumes that the accessed statistics may have changed slightly between the current and the last game

played by each player. This is achieved by slightly increasing the skill uncertainty statistic by an amount which is a configurable parameter of the system. An update process is applied to update the statistics in the light of the game outcome information (block **102**). This update process is described in more detail below. The accessed statistics are then replaced by the updated statistics (block **103**) and the resulting ranked skills of the players may then be stored (block **104**). That is, the updated skill belief statistics are stored and the previous values of those statistics may be discarded.

[0027] The game outcome information may comprise player performances and a configurable parameter, called a draw margin, is used to assess whether the player performances are close enough together for the game outcome to be considered a draw.

[0028] More detail about the update process is now given for the case of a two-player match.

[0029] The update process comprises determining the probability of the observed game outcome for given skills of the participating players and weighting it by the probability of the corresponding skill beliefs. This is done by averaging over all possible performances (weighted by their probability density values) and deriving the game outcome from the performances: The player with the highest performance is the winner; the player with the second highest performance is the first runner up, and so on. If two players' performances are very close together, then the system considers the outcome between these two players a draw. The larger the margin which defines a draw in a given league, the more likely a draw is to occur, according to the ranking system. The size of this margin is a configurable parameter of the ranking system and is adjusted based on the game mode. For example, a street race in a car racing game involving two players may almost never end in a draw if crossing the finishing line is measured with high accuracy (thus the parameter is set to almost zero). In contrast, a game between two opponents for points over a fixed period of time (like football) can easily end in a draw.

[0030] By virtue of the above weighting technique (which is based on Bayes' Law), the system arrives at a new skill belief for every player participating in the game. These skill beliefs are not Gaussian anymore. Hence, the ranking system determines the best Gaussian approximation. As a result, given players' μ values increase for each opponent they outperformed, and decreases for each opponent they lost against.

[0031] The simplest case for the ranking system update is a two-person match. Suppose we have players A(lice) and B(ob), with μ and σ values (μ_A,σ_A) and (μ_B,σ_B), respectively. Once the game has finished, the update algorithm determines the winner (Alice or Bob) and loser (Bob or Alice) and applies the following update equations:

$$\mu_{winner} \leftarrow \mu_{winner} + \sigma^2_{winner}/c * v((\mu_{winner} - \mu_{loser})/c, \epsilon/c)$$

$$\mu_{loser} \leftarrow \mu_{loser} - \sigma^2_{loser}/c * v((\mu_{winner} - \mu_{loser})/c, \epsilon/c)$$

$$\sigma^2_{winner} \leftarrow \sigma^2_{winner} * [1 - \sigma^2_{winner}/c^2 * w((\mu_{winner} - \mu_{loser})/c, \epsilon/c)]$$

$$\sigma^2_{loser} \leftarrow \sigma^2_{loser} * [1 - \sigma^2_{loser}/c^2 * w(\mu_{winner} - \mu_{loser})/c, \epsilon/c)]$$

$$c^2 = 2\beta^2 + \sigma^2_{winner} + \sigma^2_{loser}$$

[0032] In these equations, one unknown is β^2 which is the variance of the performance around the skill of each player and this is typically configured as 0.5. Moreover, ε is the aforementioned draw margin which is a configurable parameter. The functions v(.,.) and w(.,.) are given by

$$v(t, \alpha) = \frac{N(t-\alpha)}{\Phi(t-\alpha)}$$

$$w(t, \alpha) = v(t, \alpha) \cdot (v(t, \alpha) - (t-\alpha))$$

if the game ends in win and loss or

$$v(t, \alpha) = \frac{N(-\alpha - t) - N(\alpha - t)}{\Phi(\alpha - t) - \Phi(-\alpha - t)}$$

if the game ends in a draw. Where the symbols N and Φ represent the density of the Gaussian distribution function and the cumulative distribution function of the Gaussian, respectively. The symbols t and α are simply arguments to the functions. Any suitable numerical or analytic methods can be used to evaluate these functions such as those described in Press et al., Numerical Recipes in C: the Art of Scientific Computing (2d. ed.), Cambridge, Cambridge University Press, ISBN-0-521-43108-5.

[0033] There are a few observations about these update equations:

[0034] In the mean skill update equation the winner gets a multiple of $v((\mu_{winner}-\mu_{loser})/c, \epsilon/c)$ added to the mean skill and the loser gets a multiple of $v((\mu_{winner}-\mu_{loser})/c, \epsilon/c)$ subtracted from the mean skill. The weighting factors are roughly proportional to the uncertainty of the winner/loser vs. the total sum of uncertainties ($2\beta^2$ is the uncertainty due to the performance variation around the skill and $\sigma^2_{winner}+\sigma^2_{loser}$ is the uncertainty about their true skills). Note that the ranking system's update equation for the mean skill is thus not guaranteed to be zero sum.

[0035] The uncertainty of both players (regardless of win/loss/draw) is going to decrease by the factor $1-\sigma^2_{winner}/c^2 * w((\mu_{winner}-\mu_{loser})/c, \epsilon/c)$. Again, the player with the larger uncertainty gets the bigger decrease.

[0036] The change in the mean skill, $v((\mu_{winner}-\mu_{loser})/c, \epsilon c)$, and the decrease factor in the uncertainty, $1-\sigma^2_{winner}/c^2 * w((\mu_{winner}-\mu_{loser})/c, \epsilon/c)$, are close to zero if the game outcome was not surprising.

[0037] Win/Loss If the winner had the much greater mean skill relative to the total uncertainty (thus $(\mu_{winner}-\mu_{loser})>\epsilon$) then a win cannot buy the winner extra mean skill points or remove any uncertainty. The opposite is true if the game outcome was surprising: If the winner had the smaller mean skill $(\mu_{winner}-\mu_{loser})<\epsilon$), mean points proportional to $\mu_{loser}-\mu_{winner}$ get added/subtracted to/from the winner/loser.

[0038] Draw If both players had similar mean skills upfront (thus $|\mu_{winner}-\mu_{loser}|<\epsilon$) then both players are already close enough together and no mean skill point update needs to be made; hence the uncertainty is not reduced. However, if one player was thought to be much stronger by the ranking system before the game (let's say, $\mu_{winner}-\mu_{loser}>\epsilon$) then his mean skill will be decreased and the other player's mean skill will be increased which, in effect, brings their two mean skill closer together.

[0039] In the case of a team match the team's skill is assumed to be a function of the skills of the players. In a preferred embodiment, this function is the sum. The algorithm determines the sum of the skills of the two teams and uses the above two equations where $(\mu_{winner},\sigma^2_{winner})$ and $(\mu_{loser},\sigma^2_{loser})$ are the mean skills and skill variances of the winning and losing team, respectively.

[0040] The update equations for more than two teams require numerical integration. In this case the ranking system iterates two team update equations between all teams on neighbouring ranks, that is, the 1st versus the 2nd team, the 2nd team versus the 3rd team and so on. The computational complexity increases cubically for more than two teams as a result of the numerical integration required for the V and W functions. This is addressed by using factor graphs with message passing techniques to reduce the computation required in multi-team situations as now described with reference to FIG. **2**.

[0041] With reference to FIG. **2** for each player, statistics are accessed (see block **200** of FIG. **2**) describing a probability distribution that itself describes our belief about skill of that player. In a preferred embodiment we use one-dimensional Gaussian distributions to represent skill belief. By using Gaussian distributions we achieve the advantage that such distributions can be uniquely described by two statistics, the means and standard deviation, as explained above. In addition, the sum of two Gaussian random variables is itself a Gaussian random variable which enables us to simplify our computations. However, it is not essential to use Gaussian distributions to represent skill belief.

[0042] If a player has played before and we have stored skill information for that player that information is accessed. In the case of a new player we use a default belief distribution with associated default statistics, for example an initial μ of 3 and σ of 1. Any suitable default belief distribution is used.

[0043] Information about the game outcome is obtained (see block **205** of FIG. **2**) and this is used to form a factor graph (see block **201**) together with the statistics. The factor graph comprises nodes associated with particular players, those nodes being ordered on the basis of any teams and outcomes of the game. Some nodes of the factor graph are instantiated with the accessed statistic information (see block **202**). Message passing is then performed over the factor graph to update the statistics using Bayesian Inference (see block **203**). The resulting updated statistics describe our belief of the relative skill of the players (see block **204**) and the process can be repeated for further games.

[0044] More detail about the process of forming the factor graph is now given with reference to FIG. **3**. The factor graph comprises nodes **300** connected by links **301**. The nodes are either variable nodes (circles) or factor nodes (rectangles). Variable nodes represent storage locations and factor nodes represent computational units. The factor nodes read and write information to their neighbouring variable nodes according to calculation rules described later.

[0045] Each player is represented by a variable node for their skill connected to a set of nodes which relate to their skill and their performance in the particular game. In FIG. **3**, these nodes are drawn in a single column for each player, with players on the same team having nodes in adjacent columns. In the example illustrated in FIG. **3** four teams are represented, the winning team having a single player and being represented by the column on the left side of the page, **302**. The team in second place **303** comprises three players, the third place team **304** comprises two players and the last place team **305** comprises four players.

[0046] As illustrated in FIG. **3**, the factor graph is preferably acyclic. Preferably it comprises two types of nodes, variable nodes and factor nodes, and is bipartite.

[0047] The factor nodes at the top of the diagram (row **306**) are functions which access a database or other store to obtain belief distributions for each player (or use a default belief distribution in the case of a new player). These computational units feed the parameters describing the player skill belief distributions into the corresponding variable nodes. For example, in the case of Gaussian distributions there would be two parameters (two floating-point numbers) stored in each variable node. The next row of variable nodes, that is, the circular nodes **307** connected in series to the top leaf nodes, represent the player skills. These nodes each store the statistics describing the belief distribution for the associated player. The next row of factor nodes are computation units **308** which compute player performance on the basis of, in this example, player skill plus noise. That is, the skill belief distributions of units **307** are modified by increasing their variance parameters and the results are stored in the row of variable nodes **309** representing player performance. This is a deterministic computation, though it can be thought of as adding noise to the underlying random variables.

[0048] In order to obtain a representation of team performance as opposed to individual player performance the columns are combined as indicated in FIG. **3**. In the case of the second place team for example, this has three players. The performance of this team is taken to be the sum of the performance of the three players and this is indicated by the rectangular node in row **310**. Any other suitable combination rule for finding the team performance from the constituent member performances could be used. The result of that computation is stored in circular node in row **311**. Thus in row **311** there is one node per team rather than one node per player. In the case that Gaussian distributions are used the results of the summing process are also Gaussian distributions.

[0049] In a preferred embodiment as illustrated in FIG. **3** noise is added to the performance of each individual player and the player performances are then combined to form team performances. However, this is not essential. It is also possible to combine the player skills first to get a team skill, and then add noise to the team skill to obtain the team performance. The noise can be added to the individual player performances and/or to the combined team performances.

[0050] Team performance differences are represented by nodes in row **312** and each is calculated as a difference between certain nodes in the team performance layer **311** as indicated. When the game outcome provides a total ordering of the teams, then differences are calculated between consecutive teams in the ordering. In the case of a draw between teams, the teams which drew are placed in an arbitrary order amongst themselves and differences are calculated between consecutive teams in the ordering. For example, in FIG. **3** if the game outcome here were such that the middle two teams **303**, **304** drew then the nodes representing those two teams could be interchanged to produce an equally valid factor graph. Circular nodes in row **313** represent the results of the team performance difference calculation. In the case that Gaussian distributions are used the results of the difference process are also Gaussian distributions.

[0051] The bottom nodes in the graph are factor nodes which represent a calculation process encouraging the team

performance difference to be greater than the draw margin ϵ (if no draw) or less than the draw margin in absolute value (in case of a draw).

[0052] The process of message passing comprises carrying out a calculation associated with a computation node (square node in FIG. **3**) using distribution parameters from neighbouring variable nodes and passing the results to one of the neighbouring variable nodes (circular nodes in FIG. **3**). The direction of passing the results (also referred to as a processing schedule) is explained in more detail now.

[0053] The processing schedule is preferably divided into three phases: pre-processing, chain processing, and post-processing. An example pre-processing schedule is illustrated in FIG. **4**. Starting at the top factor nodes (row **306**), skill distributions are obtained from the database, or are set to default values, and computation proceeds downward along each column until the team performance row is reached (row **311**). The post processing schedule is the reverse of the pre-processing schedule but stopping at the player skills in row **307**.

[0054] After one step of pre-processing, a chain processing schedule is iterated until the belief distributions stop changing, i.e., a suitable convergence criterion has been satisfied. An example chain schedule is indicated in FIG. **4** using dotted arrows. It passes performance belief distributions back and forth between the teams until all of the performance differences satisfy the conditions imposed by the bottom factor nodes. The post-processing phase passes the performance distributions upward to obtain the new player skills. Each arrow in the processing schedules represents a non-trivial calculation and details of those calculations are given below. In the example given in FIG. **3** all the factor nodes (square calculation nodes) are exact factor nodes because they can be computed exactly except for the final indicator function nodes (referred to as order factor nodes). The order factor nodes implement an order constraint and for these nodes the associated update equations are not exact as the true factor-to-variable messages are no longer Gaussian.

[0055] The general update equations for use in carrying out the computations along the arrows in the message passing process are now given. These general update equations are tailored for use with Gaussian distributions as shown.

Factor Node Update with Gaussian Messages

[0056] Consider the factor graph of FIG. **5**.

[0057] Suppose we would like to update the message $m_{f\to x}$ and the marginal probability density p_x. Then, the general update equations are as follows:

$$m_x \to f(x) \quad \alpha \quad \frac{p_x(x)}{m_{f\to x}(x)},$$

$$m^{true}_{f\to x}(x) \quad \alpha \quad \int\int f(x,y,z)\cdot\frac{p_y(y)}{m_{f\to y}(x)}\cdot\frac{p_z(z)}{m_{f\to z}(x)}dydz,$$

$$p_x(x) \quad = \quad MM[m^{true}_{f\to x}(x)m_{x\to f}(x)],$$

$$m_{f\to x} \quad \alpha \quad \frac{p_x(x)}{m_{x\to f}(x)},$$

where MM[·] returns the distribution in the Gaussian family with the same moments as the argument and all quantities on the right are normalized to be distributions. In the following we use the exponential representation of the Gaussian, that is,

$$G(x;\tau,\pi)\alpha\ \exp(\pi x^2-2\tau x)$$

[0058] This density has the following relation to the standard density

$$G(x;\tau,\pi)\alpha\ N\left(x;\frac{\tau}{\pi},\frac{1}{\pi}\right),\ \text{or}\ N(x;\mu,\sigma^2)\alpha\ G\left(x;\frac{\mu}{\sigma^2},\frac{1}{\sigma^2}\right)$$

[0059] In the case of the exact factor nodes the update equations are given in the following table.

Factor	Update equation
Please refer to FIG. 6	$\pi_x^{new}\leftarrow\pi_x+\frac{1}{v^2}$ $\pi_x^{new}\leftarrow\tau_x+\frac{m}{v^2}$ This is exact and should only be executed once.
Please refer to FIG. 7	$\pi^{new}_{f\to x}\leftarrow a(\pi_y-\pi_{f\to y})$ $\tau^{new}_{f\to x}\leftarrow a(\tau_y-\tau_{f\to y})$ $\pi_x^{new}\leftarrow\pi_x+\pi^{new}_{f\to x}-\pi_{f\to x}$ $\tau_x^{new}\leftarrow\tau_x+\tau^{new}_{f\to x}-\tau_{f\to x}$ $a:=\frac{1}{1+c^2(\pi_y+\pi_{f\to y})}$
Please refer to FIG. 8	Please refer to FIG. 8
Please refer to FIG. 9	$\pi^{new}_{f\to x}\leftarrow\left(\sum_{j=1}^{n}\frac{a_j^2}{\pi_{y_j}-\pi_{f\to y_j}}\right)^{-1}$ $\tau^{new}_{f\to x}\leftarrow$ $\pi^{new}_{f\to x}\cdot\left(\sum_{j=1}^{n}a_j\cdot\frac{\tau_{y_i}-\tau_{f\to y_i}}{\pi_{y_j}-\pi_{f\to y_i}}\right)$ $\pi_x^{new}\leftarrow\pi_x+\pi^{new}_{f\to x}-\pi_{f\to x}$ $\tau_x^{new}\leftarrow\tau_x+\tau^{new}_{f\to x}-\tau_{f\to x}$
Please refer to FIG. 10	Please refer to FIG. 11

[0060] In the case of the order factor nodes, the update equations are given in the following table.

Factor	Update equation
Please refer to FIG. 12	$\pi_x^{new} \leftarrow \frac{C}{1-w\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}$
	$\tau_x^{new} \leftarrow \frac{d+\sqrt{C}\cdot v\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}{1-w\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}$
	$\pi_{f\to x}^{new} \leftarrow \pi_{f\to x} + \pi_x^{new} - \pi_x$
	$\tau_{f\to x}^{new} \leftarrow \tau_{f\to x} + \tau_x^{new} - \tau_x$
	$C = \pi_x - \pi_{f\to x}, d = \tau_x - \tau_{f\to x}$
Please refer to FIG. 13	$\pi_x^{new} \leftarrow \frac{C}{1-w_0\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}$
	$\tau_x^{new} \leftarrow \frac{d+\sqrt{C}\cdot v_0\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}{1-w_0\left(\frac{d}{\sqrt{c}}, \varepsilon\sqrt{C}\right)}$
	$\pi_{f\to x}^{new} \leftarrow \pi_{f\to x} + \pi_x^{new} - \pi_x$
	$\tau_{f\to x}^{new} \leftarrow \tau_{f\to x} + \tau_x^{new} - \tau_x$
	$C = \pi_x - \pi_{m_x}, d = \tau_x - \tau_{m_x}$

[0061] In the update equations set out in the tables above a represents weightings which in a preferred example are set to 1. Also, in the update equations v and w correspond to the functions v(.,.) and w(.,.) given by

$$v(t,\alpha) = \frac{N(t-\alpha)}{\Phi(t-\alpha)}$$

$$w(t,\alpha) = v(t,\alpha)\cdot(v(t,\alpha)-(t-\alpha))$$

if the game ends in win and loss or

$$v(t,\alpha) = \frac{N(-\alpha-t)-N(\alpha-t)}{\Phi(\alpha-t)-\Phi(-\alpha-t)}$$

$$w(t,\alpha) = v^2(t,\alpha) + \frac{(a-t)N(\alpha-t)-(-\alpha-t)N(-\alpha-t)}{\Phi(\alpha-t)-\Phi(-\alpha-t)}$$

if the game ends in a draw. They may be determined from the numerical approximation of a Gaussian and Gaussian cumulative distribution without using message passing.

[0062] In the example shown in FIG. **4** the message passing during the chain schedule involves order factor node updates from the nodes in row **314** to the nodes in row **313** using the update equation of the first row of the order factor node update equation table. In the case of a draw the modulus of the team performance difference is constrained to be less than or equal to a latent draw value ϵ and the update equation of the second row of the order factor node update equation table is used.

[0063] In the case of exact factor nodes, for message passing from a computation node (square node) to a single variable node (circular node) the update equations of the first row of the exact factor node update equation table is used. In the case of message passing from a computation node to two variable nodes the update equations of the second or third row of the table are used as appropriate. In the case of message passing from a computation node to three variable nodes the update equations of the fourth and fifth rows of that table are used as appropriate.

Taking into Account Past and Future Player Achievements

[0064] Most previous skill estimation systems operate in a filtering mode whereby they take into account only past game outcomes to estimate skill. This means that if player A beats an unknown player B and later it turns out that player B was in fact strong (e.g. by player B later repeatedly beating known-to-be-strong player C) these filtering-based methods are not able to retro-actively correct A's skill estimate upwards.

[0065] In many situations, information about both past and future player achievements is available. Embodiments of the present invention which use such information to improve accuracy of player skill estimates are now described. These embodiments may be thought of as comprising a smoothing of a time series of player skills which takes into account past as well as future player achievements.

An Exemplary Method

[0066] FIG. **14** is a block diagram of an example method of determining relative skills of players using both past and future game outcome information. For each player, a time series of skill belief statistics is accessed (block **1400**). For example, belief about a player's skill is modeled using a probability distribution such as a Gaussian as described above. A mean and a standard deviation value may be used to describe this probability distribution so that values of these statistics are accessed to describe a given player's skill during a specified time period. Any suitable time period may be used, such as a week, day, hour, year or other time period. If the time period is a year for example, the time series comprises, for each player, an ordered sequence of pairs of mean and standard deviation values, each pair of values representing skill belief for that player for the given year.

[0067] For each time period of the time series, information is accessed about outcomes of games played in that time period (block **1401**). For example, for year **1**, outcomes of all games in which player A took part are accessed.

[0068] Within each time period a first update process is applied. This updates the skill belief statistics on the basis of the game outcome information (block **1402**). For example, for year **1**, the game outcomes for player A are used to update skill belief statistics for player A for that year. This is repeated for all players and for all years (or other time periods).

[0069] The process also comprises applying a second update process to update the skill belief statistics both forwards and backwards in the time series (block **1403**). The first and second update processes may be integrated rather than carried out in series. The block diagram of FIG. **14** is in no way intended to indicate that the first and second update processes must be carried out in series.

[0070] The resulting skill belief statistics are then stored (block **1404**) and may be used as input to a matchmaking system, a system for displaying player rankings to end users or any other system which uses skill belief statistics.

[0071] It is also possible for the time series to comprise other statistics instead of or in addition to the mean and standard deviation values representing skill belief. Any suitable time series may be used having an ordered sequence of sets of statistic values taken at regular time intervals.

An Exemplary System

[0072] FIG. **15** is a schematic diagram of an apparatus **1500** for providing skill belief statistics of players. The apparatus has a first input **1504** which is an interface to a database **1503** or other memory storing time series of skill belief statistics for players. There is a second input **1505** for receiving game outcome information **1506**. The apparatus comprises at least a processor **1502** arranged to carry out update processes as described herein and optionally a factor graph means **1501** for representing a factor graph and using this to aid in the update processes. An output **1508** provides the skill belief statistics of players **1507** as determined by the system.

[0073] In some embodiments the update processes of FIG. **14** are carried out using message passing techniques in a factor graph representation of the time series of skill belief statistics. This reduces computational complexity and speeds up processing times. However, it is not essential to use message passing techniques for the update processes. Instead, any suitable numerical or analytic methods can be used to evaluate the update processes as described in Press et al., Numerical Recipes in C (already referred to above). This is practical where computational requirements can be met, for example, where the numbers of players and games involved is relatively low. However, in cases where hundreds of thousands of players are active over several years it is typically most practical to use the message passing techniques as described herein.

[0074] An example method of using a factor graph with message passing techniques to carry out the update processes, both within a time series interval or unit and forwards and backwards through the time series, is now described with reference to FIG. **16**.

Exemplary Method using Message Passing

[0075] For each player, a time series of skill belief statistics is accessed (block **1600**) as mentioned above with reference to FIG. **14**. Game outcome information is also accessed (block **1605**) in the same way as described with reference to FIG. **14**. A factor graph is created (block **1601**) comprising nodes associated with particular players that are ordered on the basis of the time series. These nodes are instantiated with the time series statistics (block **1602**). The factor graph also has nodes associated with the particular players and which are ordered on the basis of any teams and outcomes of games. Message passing is then performed over the factor graph to update the statistics using Bayesian inference (block **1603**). This message passing is carried out both between nodes that represent game outcomes within time series units and also forwards and backwards along the nodes representing the time series itself. The updated statistics are stored (block **1604**).

Forming the Factor Graph

[0076] More detail about the process of forming the factor graph is now given with reference to FIG. **17**. The factor graph comprises nodes **1700** connected by edges **1701**. The nodes are either variable nodes (circles) or factor nodes (rectangles). Variable nodes represent storage locations and factor nodes represent computational units. The factor nodes read and write information to their neighboring variable nodes according to calculation rules described later.

[0077] The example factor graph of FIG. **17** is for a situation involving three players with time series of skill belief statistics over a period of four years. This is an example only and in practice many more players may be represented with many more game outcomes and for time series having large sequences of values.

[0078] For each player, a time series of skill belief statistics is represented by a row of variable nodes. For example, player **1** has a time series shown in row **1702**, player **2** has a time series shown in row **1703** and player **3** has a time series shown in row **1704**. In this example, a time series unit or interval is a year so that player skill statistics for year **1** are represented in the column labeled "year 1" and so on for years **2**, **3** and **4**.

[0079] For player **1**, the time series of skill statistics from year **1** to year **4** is shown in row **1702**. It comprises skill belief statistics stored in variable nodes S_1^1, S_1^2, S_1^3, S_1^4. Those variable nodes are linked together with a factor node **1711** between each variable node. The factor nodes **1711** act to add noise to the skill belief statistics reflecting the belief that skill in year t+1 is the same as skill in year t but corrupted by noise. The skill belief statistics for this player, for year **1**, are initialized either to default values or using values read from a database as indicated by factor node **1712**. For player **2**, the time series of skill statistics is only available for years **1** through **3** as shown. For player **3**, the time series of skill statistics is available only for years **2** through **4** as shown.

[0080] Within each time series unit (in this case, each year), observed game outcomes between players are represented in the factor graph. For example, in year **1** a two player match between player **1** and player **2** has been observed where player **1** was the winner. This is represented in the factor graph using the nodes indicated by dotted line **1713**. Player **1**'s skill belief statistics for year **1** have noise added at factor node in row **1705** and the resulting performance statistics are stored at variable node P_1^1 in row **1706**. This is also done for player **2** to give variable node P_1^2 at row **1706**. The difference between these performance values is calculated at the factor node in row **1707** and the difference value stored at variable node d in row **1708**. The bottom nodes (row **1709**) in the factor graph are factor nodes which represent a calculation process encouraging the performance difference to be greater than 0 or less than 0 depending on which player was observed to be the winner.

[0081] In the example of FIG. **17** only one game is represented during year **1**. However, many more games may be represented in practice and these may be between teams of players rather than only two-player matches.

[0082] It is also possible to represent games between teams of players. For example, in year **3** a game between a team comprising players **2** and **3** against player **1** is shown. In this case, the team performance is calculated as the sum of the player performances for the team and stored at variable node **1710**.

[0083] The process of message passing comprises carrying out a calculation associated with a computation node (square node in FIG. **17**) using distribution parameters from neighboring variable nodes and passing the results to one of the neighboring variable nodes (circular nodes in FIG. **17**). The direction of passing the results (also referred to as a processing schedule) is explained in more detail now.

[0084] For each player, the processing schedule begins starting at the factor nodes **1712** from which skill distribu-

tions are obtained either from a database or are set to default values. This is represented in FIG. **17** by the solid arrows from the factor nodes such as **1712** to the first variable nodes in the time series for each player. The computation then proceeds downwards along each column stemming from the year one variable nodes until the performance row **1706** or the team performance row (in the case of teams of players) is reached. This part of the processing schedule is referred to as pre-processing.

[0085] A chain processing schedule is then iterated until the belief distributions stop changing, i.e. a suitable convergence criterion has been satisfied. An example chain schedule is indicated in FIG. **17** using dotted arrows. It passes performance belief distributions back and forth between the difference variable nodes (row **1708**) or the teams (in the case that teams of players are used) until all of the performance differences satisfy the conditions imposed by the bottom factor nodes. A post-processing phase then passes the performance distributions upward to obtain new player skills S_1^1 and S_1^2.

[0086] In some embodiments, the upward messages used in the post-processing phase are stored. Repeated updates are then made on the same game outcomes for the particular year. However, the saved upward messages are used to calculate new downward messages in order to effectively divide out the earlier upward message to avoid double counting. The process is iterated up and down the columns stemming from the particular year variable nodes until the skill statistic values remain substantially the same.

[0087] The new player skills are stored whilst retaining a record of their previous values.

[0088] Processing then proceeds one more stage along the time series for each of players **1** and **2** by adding noise to reach variable nodes S_1^2 and S_2^2. The messages used to add the noise are also stored. Variable node S_2^3 is also initialized as described above by reading statistics from a database or setting default values.

[0089] Computation follows down the columns stemming from the year **2** variable nodes and continues with pre-processing, chain processing and post processing as described above. Iteration up and down the columns may be carried out as described above until the player skill statistics for that year reach convergence. This gives new player skills S_1^2, S_2^2 and S_3^2.

[0090] Again processing proceeds one more step along the time series for each of the players and then again within the columns stemming from the year **3** variable nodes. This process repeats until the end of each time series is reached.

[0091] Processing now makes its way backwards along each time series, again with processing within the columns stemming from each year at each time step in the series. However, it is necessary to avoid duplicating the effect of the previous steps in the processing chain. In order to do this, when the updates are made in the backwards direction along each time series, the effect of the previous forwards updates are removed. This is achieved using the records that were stored of the messages used and the variable values.

[0092] For example, in the backwards pass along each time series, the stored messages previously used in the forwards pass are used to calculate new messages for use in the backwards pass.

[0093] The procedure is iterated forward and backward along the time series of skill statistics for each player until the skill statistics do not change significantly. The backward passes make it possible to propagate information from the future into the past.

[0094] More detail about the processing stages is now given with reference to an example involving two player games. However, the methods described can equally be used for games with any number of teams having one or multiple players per team. A series of game outcomes between two players i and j in year t is denoted by $y_{ij}^t(k) \epsilon \{+1,-1,0\}$ where $k \epsilon \{1, \ldots, K_{ij}^t\}$ denotes the number of game outcomes available for that pair of players in that year. y=+1 if player i wins, y=−1 if player j wins and y=0 in case of a draw.

[0095] The update process for game outcomes within a time series step (in this case a year) involves going through the game outcomes y_{ij}^t within a year t several times until convergence. The update for a game outcome $y_{ij}^t(k)$ is performed as described above with reference to FIGS. **3** and **4** but saving the upward messages $m_{f(p_{ij}^t(k),s_i^t)\to s_i^t}(s_i^t)$ which describe the effect of the updated performance $p_{ij}^t(k)$ on the underlying skill s_i^t. When game outcome $y_{ij}^t(k)$ comes up for update again, the new downward message $m_{f(p_{ij}^t(k),s_i^t)\to p_{ij}^t(k)}(p_{ij}^t(k))$ is calculated by

$$m_{f(p_{ij}^t(k),s_i^t)\to p_{ij}^t(k)}(p_{ij}^t(k)) = \int_{-\infty}^{\infty} f(p_{ij}^t(k), s_i^t) \frac{p(s_i^t)}{m_{f(p_{ij}^t(k),s_i^t)\to s_i^t}(s_i^t)} ds_i^t,$$

thus effectively dividing out the earlier upward message to avoid double counting. The integral above may be evaluated since the messages as well as the marginals $p(s_i^t)$ have been assumed Gaussian. The new downward message serves as the effective prior belief on the performance $p_i^t(k)$. At convergence, the dependency of the inferred skills on the order of game outcome vanishes.

[0096] The method also involves repeatedly smoothing forward and backward in time. During the first forward pass along the time series of each player the forward messages $m_{f(s_i^{t-i},s_i^t)\to s_i^t}(s_i^t)$ are stored. They represent the influence of skill estimate s_i^{t-1} time t−1 on skill estimate s_i^t at time t. In the backward pass, these messages are then used to calculate the new backward messages $m_{f(s_i^{t-1},s_i^t)\to s_i^{t-1}}(s_i^{t-1})$, which effectively serve as the new prior for time step t−1,

$$m_{f(s_i^{t-1},s_i^t)\to s_i^{t-1}}(s_i^{t-1}) = \int_{-\infty}^{\infty} f(s_i^{t-1}, s_i^t) \frac{p(s_i^t)}{m_{f(s_i^{t-1},s_i^t)\to s_i^t}(s_i^t)} ds_i^t,$$

[0097] This procedure is repeated forward and backward along the time series of skills until convergence. The backward passes make it possible to propagate information from the future into the past.

[0098] By making repeated updates on game outcomes within each time series step then the dependency of the inferred skills on the order of game outcomes is removed. This improves the accuracy of the skill estimates.

[0099] Also, by propagating information both forward and backwards along the time series accuracy of skill estimates is further improved. Previous systems have not been able to propagate information backwards in time.

[0100] The general update equations for use in carrying out the computations in the message passing process are as

described above with reference to FIGS. **3** and **4** but with the new downward or backward messages being calculated as described above.

[0101] In some embodiments average skill belief statistics per time series interval, over all players are determined. These average values may then be used to initialize skill belief statistics for previously unobserved players; that is, in place of the default statistic values mentioned above.

Individual Draw Margins

[0102] Previous player skill estimation systems have assumed a fixed draw probability per game mode or type and have required an operator or developer to pre-configure this parameter, also referred to as a draw margin. However, it is recognized herein that the probability of draw may be positively correlated with playing skill and that it may vary considerably across individual players.

[0103] A player skill estimation process is now described which determines a draw margin parameter value for each individual player in addition to player skill estimates.

[0104] As shown in FIG. **18** skill belief statistics are accessed for each player (block **1800**) and this is carried out in a similar manner as described above with reference to FIG. **1**. The skill belief statistics comprise for example, a mean and a standard deviation for each player describing a Gaussian probability distribution representing belief about skill of that player. For each player, draw margin belief statistics are also accessed (block **1801**). A probability distribution is used to represent belief about the draw margin for each player. This can be thought of as a distribution representing belief about a player's ability to force a draw. Again this probability distribution may be a Gaussian although this is not essential; any suitable probability distribution may be used. Statistics describing the draw margin belief distribution may be a mean and a standard deviation in the case of a Gaussian distribution although any suitable statistics may be used. Thus for each player, four statistics are available in some embodiments. These four statistics comprise a mean skill belief, a mean skill belief variance, a mean draw margin belief and a mean draw margin belief variance.

[0105] Game outcome information is accessed (block **1802**) and used to update the statistics using a Bayesian inference process (block **1803**). The updated statistics are stored (blocks **1804** and **1805**) and may be used in any suitable application such as a matchmaking process, a player ranking process or any other suitable application.

[0106] The Bayesian inference process is carried out using similar methods to those described above and may either involve using factor graphs and message passing or may use any suitable numerical or analytic methods.

[0107] For example, suppose each player i at every time-step t is characterized by an unknown skill $s_i^t \in \Re$ and a player-specific draw margin $\epsilon_i^t > 0$. Performances $p_{ij}^k(k)$ and $p_{ji}^k(k)$ are drawn according to $p(p_{ij}^t(k)|s_i^t)=N(p_{ij}^t(k);s_i^t,\beta^2)$. In this model a game outcome $y_{ij}^t(k)$ between players i and j at time t is generated as follows:

$$y_{ij}^t(k)=\begin{cases}+1 & \text{if} \quad p_{ij}^t(k) > p_{ji}^t(k)+\varepsilon_j^t \\ -1 & \text{if} \quad p_{ji}^t(k) > p_{ij}^t(k)+\varepsilon_i^t \\ 0 & \text{if} \quad -\varepsilon_i \le p_{ij}(k)-p_{ji}^t(k) \le \varepsilon_j\end{cases}$$

Where y=+1 if player i wins, y=−1 if player j wins and y=0 in the case of a draw. A factorizing Gaussian distribution is assumed for the player-specific draw margins $p(\epsilon_i^0)=N(\epsilon_i^0; v_0,\zeta_0^2)$ and a Gaussian drift of draw margins between time steps given by $p(\epsilon_i^t|\epsilon_i^{t-1})=N(\epsilon^t;\epsilon^{t-1};\zeta^2)$. A factorizing Gaussian prior $p(s_i^0)=N(s_i^0;\mu_0,\sigma_0^2)$ is assumed over skills and a Gaussian drift of skills between time steps given by $p(s_i^t|s_i^{t-1})=N(s^t;s^{t-1};\tau^2)$.

[0108] In an example, the update process is carried out using factor graphs and message passing. FIG. **19** shows an example factor graph for use in the case of a win or a loss in a two-player game and FIG. **20** shows an example factor graph for use in the case of a draw in a two-player game. The factor graphs may be extended for games involving teams of multiple players.

[0109] The factor graph of FIG. **19** comprises a variable node for the skill statistics s^t_W of the winning player at time t, a variable node for the skill statistics s^t_L of the losing player at time t, and a variable node for the draw margin statistics ε^t_L of the losing player at time t. The draw margin statistics of the losing player at time t are forced to be positive by a factor node **1900**.

[0110] Factor nodes **1901** are used to add noise to the skill statistics of the winner and loser to obtain performance statistics p_W and p_L as indicated. The difference between the performance of the winner and loser is forced to be greater than the draw margin of the loser plus a winning threshold u_L of the loser. Message passing occurs in a similar manner to that described above with reference to FIGS. **3** and **4**.

[0111] FIG. **20** shows an example factor graph for use in the case of a draw in a two-player game between player i and player j. The factor graph comprises a variable node for the skill statistics s^t_i of player i at time t, a variable node for the skill statistics s^t_j of player j at time t, a variable node for the draw margin statistics ε^t_i of player i at time t and a variable node for the draw margin statistics ε^t_j of player j at time t. The draw margin statistics of the players at time t are forced to be positive by a factor nodes **2000**.

[0112] Factor nodes **2001** are used to add noise to the skill statistics of the players to obtain performance statistics p_i and p_j as indicated. The factor graph then encodes a situation where player i beats player j only if i's performance is bigger than j's performance by more than j's own ability at forcing a draw. Also, player j beats player i only if j's performance is bigger than i's performance by more than i's ability to force a draw. Message passing occurs in a similar manner to that described above with reference to FIGS. **3** and **4**.

[0113] In some embodiments average draw margin belief statistics per time series interval, over all players are determined. These average values may then be used to initialize draw margin belief statistics for previously unobserved players; that is, in place of the default statistic values mentioned above.

[0114] It is also possible to combine the processes described above whereby individual player draw margin statistics are determined with the processes described above whereby the update process enables both past and future player achievements to be taken into account. For example, inference may be carried out both within a given year t (or other time series interval) as well as across years (or other time series interval) in a forwards and backwards manner.

Exemplary Computing-Based Device

[0115] FIG. **21** illustrates various components of an exemplary computing-based device **2100** which may be imple-

mented as any form of a computing and/or electronic device, and in which embodiments of a system for estimating skills and/or draw margins of players may be implemented.

[0116] The computing-based device **2100** comprises one or more inputs **21 02** which are of any suitable type for receiving media content, Internet Protocol (IP) input, information about outcomes of games between players, skill statistics and draw margin statistics. The device also comprises communication interface **2109** for interfacing with a game matchmaking system, game leader board system or other system requiring player skill estimates and/or player draw margin estimates.

[0117] Computing-based device **2100** also comprises one or more processors **2101** which may be microprocessors, controllers or any other suitable type of processors for processing computing executable instructions to control the operation of the device in order to estimate skill statistics of players and/or draw margin statistics of players. Platform software comprising an operating system **2106** or any other suitable platform software may be provided at the computing-based device to enable application software **2107** to be executed on the device.

[0118] The computer executable instructions may be provided using any computer-readable media, such as memory **2105**. The memory is of any suitable type such as random access memory (RAM), a disk storage device of any type such as a magnetic or optical storage device, a hard disk drive, or a CD, DVD or other disc drive. Flash memory, EPROM or EEPROM may also be used. A data store **2108** may also be provided for storing skill statistics and/or draw margin statistics.

[0119] An output **2103** is also provided such as an audio and/or video output to a display system integral with or in communication with the computing-based device. The output **2103** may also provide skill statistics and/or draw margin statistics in any suitable form such as by writing to a file, disk or other removable storage medium. A display interface **2104** may be provided to enable a graphical user interface, or other display interface although this is not essential.

[0120] The term 'computer' is used herein to refer to any device with processing capability such that it can execute instructions. Those skilled in the art will realize that such processing capabilities are incorporated into many different devices and therefore the term 'computer' includes PCs, servers, mobile telephones, personal digital assistants and many other devices.

[0121] The methods described herein may be performed by software in machine readable form on a storage medium. The software can be suitable for execution on a parallel processor or a serial processor such that the method steps may be carried out in any suitable order, or simultaneously.

[0122] This acknowledges that software can be a valuable, separately tradable commodity. It is intended to encompass software, which runs on or controls "dumb" or standard hardware, to carry out the desired functions. It is also intended to encompass software which "describes" or defines the configuration of hardware, such as HDL (hardware description language) software, as is used for designing silicon chips, or for configuring universal programmable chips, to carry out desired functions.

[0123] Those skilled in the art will realize that storage devices utilized to store program instructions can be distributed across a network. For example, a remote computer may store an example of the process described as software. A local or terminal computer may access the remote computer and download a part or all of the software to run the program. Alternatively, the local computer may download pieces of the software as needed, or execute some software instructions at the local terminal and some at the remote computer (or computer network). Those skilled in the art will also realize that by utilizing conventional techniques known to those skilled in the art that all, or a portion of the software instructions may be carried out by a dedicated circuit, such as a DSP, programmable logic array, or the like.

[0124] Any range or device value given herein may be extended or altered without losing the effect sought, as will be apparent to the skilled person.

[0125] It will be understood that the benefits and advantages described above may relate to one embodiment or may relate to several embodiments. It will further be understood that reference to 'an' item refers to one or more of those items.

[0126] The steps of the methods described herein may be carried out in any suitable order, or simultaneously where appropriate. Additionally, individual blocks may be deleted from any of the methods without departing from the spirit and scope of the subject matter described herein. Aspects of any of the examples described above may be combined with aspects of any of the other examples described to form further examples without losing the effect sought.

[0127] It will be understood that the above description of a preferred embodiment is given by way of example only and that various modifications may be made by those skilled in the art. The above specification, examples and data provide a complete description of the structure and use of exemplary embodiments of the invention. Although various embodiments of the invention have been described above with a certain degree of particularity, or with reference to one or more individual embodiments, those skilled in the art could make numerous alterations to the disclosed embodiments without departing from the spirit or scope of this invention.

1. A method comprising:
receiving information about an outcome of a game between at least a first player opposing a second player;
receiving, for each player, skill statistics associated with a distribution representing belief about skill of that player;
receiving, for each player, draw margin statistics associated with a distribution representing belief about ability of that player to force a draw;
updating the statistics on the basis of the received information about the game outcome.

2. A method as claimed in claim **1** wherein the updating process comprises using a Bayesian inference process.

3. A method as claimed in claim **1** wherein the step of receiving information about the game outcome comprises receiving an outcome of a game between at least a first team opposing a second team, the first team comprising a plurality of players and the second team comprising at least one other player.

4. A method as claimed in claim **1** wherein the updating process comprises forming a factor graph using the received information about the game outcome and carrying out a message passing process in the factor graph.

5. A method as claimed in claim **1** wherein the skill statistics comprise an individual mean and an individual variance for each player.

6. A method as claimed in claim **1** wherein the draw margin statistics comprise an individual mean and an individual variance for each player.

7. A method as claimed in claim 1 which further comprises receiving information about a player proposing a game and selecting at least one other player for the proposed game on the basis of draw margin statistics determined for that at least one other player.

8. A method as claimed in claim 1 wherein the step of receiving information about a game outcome comprises receiving information about a plurality of game outcomes; and wherein the process of receiving skill statistics and draw margin statistics comprises receiving a time series of skill statistics and a time series of draw margin statistics for each player.

9. A method as claimed in claim 8 which further comprises updating both time series of statistics both forwards and backwards in those time series on the basis of the game outcome information.

10. A method as claimed in claim 8 which further comprises determining average skill statistic values over all players for each time series interval.

11. A method as claimed in claim 8 which further comprises determining average draw margin statistic values over all players for each time series interval.

12. One or more computer readable media containing computer executable instructions that, when implemented, perform a method comprising:

receiving information about an outcome of a game between at least a first player opposing a second player;

receiving, for each player, skill statistics associated with a distribution representing belief about skill of that player;

receiving, for each player, draw margin statistics associated with a distribution representing belief about ability of that player to force a draw;

updating the draw margin statistics on the basis of the received information about the game outcome.

13. One or more computer readable media as claimed in claim 12 wherein the updating process comprises using a Bayesian inference process.

14. One or more computer readable media as claimed in claim 12 wherein the updating process comprises forming a factor graph using the received information about the game outcome and carrying out a message passing process in the factor graph.

15. One or more computer readable media as claimed in claim 12 wherein the skill statistics comprise an individual mean and an individual variance for each player.

16. One or more computer readable media as claimed in claim 12 wherein the draw margin statistics comprise an individual mean and an individual variance for each player

17. One or more computer readable media as claimed in claim 12 wherein the step of receiving information about a game outcome comprises receiving information about a plurality of game outcomes; and wherein the process of receiving skill statistics and draw margin statistics comprises receiving a time series of skill statistics and a time series of draw margin statistics for each player.

18. One or more computer readable media as claimed in claim 17 wherein the updating process further comprises updating both time series of statistics both forwards and backwards in those time series on the basis of the game outcome information.

19. One or more computer readable media containing computer executable instructions that, when implemented, perform a method comprising:

receiving information about an outcome of a game between at least a first player opposing a second player;

receiving, for each player, skill statistics associated with a distribution representing belief about skill of that player;

receiving, for each player, draw margin statistics associated with a distribution representing belief about ability of that player to force a draw;

updating the statistics on the basis of the received information about the game outcome; and

wherein the step of receiving information about a game outcome comprises receiving information about a plurality of game outcomes; and wherein the process of receiving skill statistics and draw margin statistics comprises receiving a time series of skill statistics and a time series of draw margin statistics for each player.

20. One or more computer readable media as claimed in claim 19 wherein the updating process further comprises updating both time series of statistics both forwards and backwards in those time series on the basis of the game outcome information.

* * * * *