ADVANCED INFO IN LUCASCHESS 8.0

LucasChess 8.0 is trying, likely for the very first time in the history of the chess softwares, to implement a numerical method to show and tell its users some advanced abstract and concrete informations about the position.
This is a very hard target: it is both an intellectual challenge and it is also very useful for the users, because they will understand more about the game than ever before.
Surely in the process of this development, we will change many times the way we "calculate" those information we have spoken about.
In order to achieve that LucasChess needs to know, after every and each move, a certain number of things, both by asking them to the running engine and by calculating itself.

Here is a list of original variables (with a sort of evocative names):
xplm: number of half moves (plies) already played (range: from 1 to infinite, naturally an integer).
xpie: number of pieces still alive in the board (range: from 2 to 32, an integer).
xpiec: number of pieces belonging to the side to move still alive in the board (range: from 1 to 16, an integer).
xmat: a sum of all the material stile alive in the board (range: from 6 to 100 or more; it is a fractional number where each king contributes with 3 points, each queen with 9.9, each rook with 5.5, each bishop with 3.5, each knight with 3.1 and each pawn with 1.0).
xpow: a sum of all the material belonging to the side to move still alive in the board (range: from 3 to 50 or more, a fractional number).
xmov: number of legal moves that is possible to play in a specific position, regardless of their precision and efficiency (range: from 1 to 50 or more, an integer).
xgmo: good moves available, within 100 centipawns from the best with a different weighted contribution, that is greater for the best moves and lesser for the worst ones (range: typically from 1 to 10 or even more, a fractionl number).
xeval: value that the engine has calculated to evaluate the position (range: a fractional number that depends by the engine; each engine has its own evaluation and therefore that leads to different values according to the running engine).

Some other variables are derived, meaning that they are just a proper combination of the originl ones (they all are fractional numbers, expressed as a percentage):

xcompl: the value that shows how complex is a position (range: typically from 0 to 80 or 90, in all our simulations it very hardly overtook 100).
xmlr: a value that tells how likely will occurr the victory of the winning side (range: it varies from 0 to +100).
xemo: a value to show how efficient is the mobility of a player (range: from 0 to 100).
xnar: a value that shows how narrow and/or crowded is a position (range: from 0 to 200, but very often it stays below 100).
xact: a value that shows how active are the pieces of the side to move(range: from 0 to 200, but very often i stays below 100).
xext: a value that shows how great is the tendency of a player to exchange pieces, or in other words, his/her tendency to simplify (range: from slightly negative values to 100 or even more).
xgst: a value that tells which stage of the game we are in (there are five possibilities: opening, transition to middlegame, middlegame, transition to endgame, endgame).

COMPLEXITY

There is no unanimously accepted idea, currently, about the way to show how complex is a chess position, and likely will never exist a safe way to get it.
In the latest fritz gui there is a tachimeter that shows how that but we are not sure that it works the way we have imagined (naturally this doesnt mean that we are right and chessbase is wrong).
We should also consider the fact that what is complex for an engine (for example to follow a long plan) might not be for a human and naturally it is true the opposite as well (for example a checkmate in 7).
We have decided to follow a human-oriented path.
For *complexity* we mean a value that tells the user how difficult is to find, in a certain position, the best move.

Here is a very good approximation of our idea:

$$xcompl = \frac{xgmo * xmov * xpie}{200} * \frac{xmat}{2 * xpow}$$

Main concepts involved in the building of the algorythm:
1)**xpie** has been put in the numerator because we assume that in a certain position, the *complexity* must be higher when the number of pieces is higher.
Even if a piece cannot move yet in a certain imaginary position, it has the "potential" to free his path or instead to be sadly captured; the more pieces the user has to consider in his thoughts, the more complex the position must somehow be.
2)**xmov** has been put in the numerator because we assume that in a certain position, the *complexity* must be higher when the number of legal moves is higher.
Strong players have a developped "sixth sense" that helps them to discard unuseful moves very quickly (although sometimes these apparently stupid moves hide amazing combinations), but neverthless it appears very logical to assess that the more legal moves a position contains the more difficult is the choice of the best one.
3)**xgmo** has been put in the numerator because we assume that in a certain position, the *complexity* must be higher when the number of good moves is higher.
And not seldom the choice of a move among three or four, apparently not so different one another, represents the boundary between a win and a draw, and even sometimes a loss!
We might say that **gmo**'s aim is to simulate a kind of more refined "sixth

sense": only acceptable moves (within 100 centipawns from the supposed best one) give contribution to this variable.
4)**xeval** is put in the denominator because obviously a very good position (therefore an high **xeval**) is easy to play and many moves can help to win; then an high **xeval** must logically be equivalent to a little complexity degree.
5)in the main denominator we have put a constant k=200, in order to get results (for xcompl) that somehow will always vary from 0 to 100 (or just little more, but very rarely).
6)the factor **(xmat/2xpow)** is to be considered and called as material balance factor: for example, values higher than 1 show that the side to move suffers of a material deficit; it may be agreeable that when a player is materially down his play becomes more difficult.

About users, in our opinion, a value of *complexity* that varies from zero or almost zero to around 100 is very easy to understand; though it has no mathematical meaning at all, it comes natural associating such values with the concept of percantage.
And actually the formula has been designed to achieve that.
LucasChess shows (in the abstract kibitzer) the value itself as a percentage and aside the value a text string that is related to the value:

0 ≤ **xcompl** <5	very low
5 ≤ **xcompl** <15	low
15 ≤ **xcompl** < 35	moderate
35 ≤ **xcompl** < 55	high
55 ≤ **xcompl** < 85	very high
85 ≤ **xcompl** <+infinite	extreme

WIN PROBABILITY

Another useful information given by LucasChess is the *win probability* (of the better side) in a certain position.
This is expecially useful in order to tell the user how good is his/her technique because if this value keeps on increasing during the game after the user has got a good position, it means that the user is beating the opponent engine displaying a very consistent and precise play.
It also can teach beginners about some endgames, if they are won or draw with absolute certainty (as long as it is used a modern engine with long thinking time) or about some positions that are impossible to win.

Our idea:

$$xmlr = \left| 100 * \tanh\left(\frac{xeval}{2 * xmat}\right) \right|$$

All concepts involved in the building of the algorythm are:
1)**xeval** has been put in the numerator because we assume that in a certain position the higher is **eval** the greater probabilities to win belong to the winning side.
It's a good approximation that an **eval**=+600 must mean that position is surely always won (mind that top engines recognize draw endgames like NNK vs K with no need to go very deep).
For **eval**=+600 and very low material amount we have **mlr**≈100 (that means that LC will show "100.0% win percentage for white").
For **eval**=+300 and just few pieces exchanged we might have **mlr**≈95; it means that LucasChess recognizes a very high probability of a victory but the worst side still has a lot of pieces to create a counterplay.
Naturally if **eval**=0 it might be highly speculative to assign a winning percentage to any of the two players; it is a rare situation in opening and middlegame, but enough common in endgames where indeed there are many positions impossible to win (that means that LucasChess will show "0% win percentage").
2)**(2xmat)** is a correcting factor: it takes into account the remaining material; it is an acceptable assumption to think that the lower is the material on the board, the more likely is the victory of the winning side because the other player has (likely) less firepower to generate a counterplay.
3)*tanh* is the trigonometric function hyperbolic tangent; naturally x is not an angle but we have discovered after hours of manual tests that it works wonderfully.

We have even run hundreds of test-games (where the same engine played both white and black) starting from fixed position, with known win probability.
The actual results and the forecasts given by our algorythm are extremely similar, much more than we ever expected.

LucasChess will show this **mlr** as percentage to win for the winning side, and aside an adjective related; exactly the same as in complexity.

EFFICIENT MOBILITY

It is possible that the user wishes to know if his/her position offers many acceptable resources or not: in other words LucasChess wants to show how efficient is the mobility at disposal of a player.
Somehow it is possible to say that *efficient mobility* measures how forcing is the nature of the position, from the point of view of the side with the right to move.
This is an information not to undervalue at all.
Moreover, there is no safe correlation between *complexity* and *efficient mobility*, because the two data really measure two very different things: a player might have a great mobility but his/her position is not necessarily either simple or complex.

A simple and yet effective formula can be:

$$xemo = 100 * \frac{xgmo-1}{xmov}$$

Main concepts involved in the building of the algorythm:
1)**xgmo** has been put in the numerator because it is logical that the more good moves there are in a certain position, the more efficient is supposed to be the mobility of a player.
2)**xmov** has been put in the denominator because it is like that that we get a ratio (read... percentage) of good moves.
3)**xgmo-1** in the numerator is put (instead of a simple **xgmo**) in order to exalt the importance of good and not forced (read... efficient) mobility: for example when there is only one legal move to make, or also when there is only one saving resource in the position.

While *complexity* and *win probability* have been purposely designed to appear as a percentage, *efficient mobility* is instead very close to be a pure percentage by its nature.
With these facts, **xemo** shows the percentage of good moves compared to the number of moves not immediately losing; notice that "good moves" are not necessarily moves that leads to advantage, they are simply the best resources of a player in that particular position.
Actually, efficient mobility can be used to spot a combination hidden in the position: when a player must not face any immediate threat and his/her position is rather active and dynamic, a very low value of **xemo** is not a bad sign!
On the contrary it means that likely there is move that keeps or even increase the advantage.

The same applies when a player is in troubles: a very low **xemo** just tell that he/she has likely only one saving resource.

LucasChess will show *efficient mobility* as a percentage of efficiency of the whole mobility, and aside an adjective related.

NARROWNESS

Another point of view about any position that LucasChess offers to its users is the *narrowness*.
This parameter tells, in our hopes, how crowded and/or narrow is a position, in brief its nature: this can be particularly useful in order to show how the properties of the pieces can vary according to this feature of the position.

Our idea is:

$$xnar = 10 * \frac{xpie}{xgmo^{0.5} * xmov^{0.5}} * \frac{xpie}{2 * xpiec} * \frac{xmat}{2 * xpow}$$

Main concepts involved in the building of the algorythm:
1)**xpie** has been put in the numerator because the number of pieces is the main factor that makes a position wide or narrow; **xpie** is a favouring factor of *narrowness*.
2)**xgmo** has been put in the denominator because an high number of good moves logically is related to a certain mobility; **xgmo** must be a contrasting factor of *narrowness*.
3)**xmov** has been put in the denominator because an high number of legal moves, good or bad, show us somehow how many pieces can actually move; again a contrasting factor of *narrowness*.
Somebody might argue that it was better to use only **xmov** or only **xgmo**: we disagree at all because if a player has a restricted and narrow position he/she cant seriously think to give a way a pawn or a whole piece in order to free his/her play (unless the engine doesnt show that the he/she gets big compensation).
Therefore not all moves that somehow open lines are acceptable to decrease *narrowness*, and that pushes for the sole use of **xgmo**.
But neverthless, when a player has 30 legal moves at his/her disposal, it is somehow "wrong" to assert that he/she has a very narrow position.
We believe that a denominator $\mathbf{(xgmo*xmov)^{0.5}}$ is a good compromise.
3)the factor **(xpie/2xpiec)** is to be considered and called as a numerical balance factor: it is a ratio involving the total number of pieces (numerator) and just the pieces of the side to move (denominator).
All values beyond 1 mean that the waiting side has more pieces than the side to move: this is a situation that somehow increases *narrowness*.
4)the factor **(xmat/2xpow)** basically exploits the same concepts than **(xpie/2xpiec)**, with the difference that here we compare the material.
It is logical in our opinion that a material advantage must somehow imply a lower narrowness, while a material disadvantage must lead to

an higher narrowness.

Narrowness and *efficient mobility* are actually enough close concepts but not identical: by the former LucasChess shows how efficiently the pieces occupy the board, by the latter LucasChess shows the percentage of moves that are not forced.
It is possibly to assert that *narrowness* is a semi-statical feature of the position while *efficient mobility* is a dynamic one.

LucasChess will show *narrowness* as a percentage and aside an adjective related.

PIECES ACTIVITY

Very often you may read in chess books or articles that you must maximize the activity of your pieces, that you must increase the activity of your worst piece if there is nothing else forcing and so on...
But what is exacly the activity of a piece?
In our opinion, and this is the way that LucasChess deals with the matter, we define *pieces activity* as the ability of all pieces belonging to a player to generate good moves and threats.

A good approximation of our idea for *pieces activity*:

$$xact = 10 * \frac{xgmo^{0.5} * xmov^{0.5}}{xpiec} * \frac{2 * xpiec}{xpie} * \frac{2 * xpow}{xmat}$$

To build the algorythm we have done these assumptions:
1)**$(xgmo*xmov)^{0.5}$** has been put in the numerator because a larger number of good moves must be directly proportional to the whole activity of a player, but also **xmov** may contribute positively because a large number of legal moves must somehow be considered an helping factor.
2)**xpiec** belongs to the denominator because we have indeed defined pieces activity as the whole activity of a player related to the number of his/her pieces.
3)both factors **(2xpiec/xpie)** and **(2xpow/xmat)**, though in inverted shape compared to narrowness algorythm, have the same meaning we have discussed before: they measure how much the material balance and numerical balance may affect the *pieces activity*.
Naturally we have inverted their shape here, because *pieces activity* concerns what is almost exactly the opposite than *narrowness*: the side to move often needs to have more pieces than his/her opponent and also more material to be more active.
4)*pieces activity* may somehow seem identical to *efficient mobility*, and indeed it is rather similar but absolutely not coincident; the difference is that while *efficient mobility* measures what is the ratio of good moves compared to all moves available to the player (therefore telling us how forced is the play in the position), *pieces activity* instead measures how many good moves per piece we have at our disposal.
We have to say also that *pieces activity* and narrowness are not exactly the opposite: the former is a measure of the dynamism of our pieces, while the latter is a semi-static feature of the position.

LucasChess will show *pieces activity* of the side to move as a percentage and an adjective related.

SIMPLIFICATION

LucasChess offers its users the possibility to take note of the different tendency of white and black to simplification, that normally involves captures and promotions.
This information might not be as useful as the previous ones, but neverthless give us a an alternative point of view about a game, and it is also a way to characterize the playstyle of a player.

Our idea of a proper algorythm is:

$$xext = 100 * \left(\frac{16}{xpiec} * \frac{45}{xpow}\right)^{0.5} - xplm - 100$$

To build the algorythm we have considered these issues:
1)the factor $\mathbf{(16/xpiec)^{0.5}}$ measures how fast pieces (belonging to the side to move) disappear from the board; it is actually an extinction factor.
2)the factor $\mathbf{(45.1/xpow)^{0.5}}$ measures how fast material (belonging to the side to move) disappears from the board; it is a complementary extinction factor.
Notice, however, that by using the second factor we have added a refined point of view about this issue: while the first factor doesnt consider what type of piece is being exchanged, the second one does! To exchange queens affects *simplification* much more than capturing some pawns each other!
3)**(-100-xplm)** is a correcting element, to take into considerations that if a position remains stable in terms of captures or promotions then it means that the players are not trying to simplify matters, but instead the contrary: indeed this is an increasing negative term in the algorythm.

As general rule, it can be said the a low *simplification* is typical of attacking players who dont like to exchange too many pieces too soon. The value of **xext** generally increases along the game, with higher rythm after an exchange; but its value will decrease after a promotion: this is optimal, because a promotion adds complications to the position.

LucasChess will show the *simplification* of the side to move as a percentage and aside an adjective related.

GAME STAGE

LucasChess is able to show which stage of the game we are in.
We have gone beyond the traditional classification in three stages (opening, middlegame and endgame) and indeed the algorythm we have developped is able to recognize two more stages: transition to middlegame and transition to endgame.
It must be told, neverthless, that even among the best known theoreticians there is no agreement at all about where/when/how a stage ends and the next starts; not only as a general rule, but even concerning a specific opening.
Therefore this information should be handled carefully; it is very possible that, in the future, we will melt this algorythm with the standard opening book given and with eventual endgame bases.

We think neverthless that this is a very good approximation:

$$gst = \frac{xpie * xmat}{3 * (xplm + 0.001)}$$

Main concepts involved in the building of the algorythm:
1)**(xpie*xmat)** is a mixed material factor: it takes into consideration both the number of pieces and the amount of material remaining; basically it is correct to assert that all stages in a chess games are defined by the number of pieces **(xpie)** and its type **(xmat)**.
2)**(xplm+0.001)** gives us the opportunity to parametrize material in relation with the number of moves played.
Again we use a constant **k=0.001** in order to avoid the denominator to be zero (as it is before the game actually starts).
Though the algorythm seems quite simplistic (it is, actually), it works surprisingly well.
A further development might be adding a correcting factor that takes into account big material imbalances (for example mating attacks in the early opening).

LucasChess will shows which stage of the game we are in, according to the following table:

0 ≤ **gst** <5	endgame
5 ≤ **gst** <10	transition to endgame
10 ≤ **gst** < 40	middlegame
40 ≤ **gst** < 50	transition to middlegame
50 ≤ **gst** ≈ 962000	opening