An Algorithm for Automatic Machine Learning of the in Chess Game

Vladan Vuckovic
Faculty of Electronic Engineering, ul. Aleksandra Medvedeva 14, Niš
Serbia
{vladan.vuckovic@elfak.ni.ac.rs}

ABSTRACT: *We have designed a system to generate an algorithm for automatic machine learning of the game in chess. We defined the transposition database and conversion of the middle-code. We have obtained results and the outcome is implemented and finally verified.*

Keywords: Theory of Logic Games, Computer Chess, Algorithms, Opening Databases, Machine Learning

Received: 2 October 2021, Revised 2 March 2021, Accepted 9 March 2021

DOI: 10.6025/jdp/2021/11/2/49-55

1. Introduction

In the course of working with transposition base openings, the main problems that arise are insetting with the opening base to the main program [1], [2]. The use of standard base openings, or bases that are generated by other programs, can lead to instability at the game in the opening up and losing of important paces in the game [3], [4]. This effect is known from earlier even, in the initial versions opening bases, when grandmaster parties were used to build a base [5]. Programs played the stored opening moves, but at the moment of leaving the base, they wandered in the game plan, because they did not know the basic ideas and principles from particular opening systems. Not understanding the purpose of the particular moves in the opening, if often happened that there had been unnecessary maneuvering of certain figures, or multiple playing with moves in the same opening, which according to the principles of the game of chess is considered wrong. Simply put, in the situation of the lack of knowledge of the general principles of opening line, the program was trying to optimize the position of its figures according to its evaluation function. By using methods of incremental learning (temporal learning), it is possible to conceive of several methods of automatic learning of various parameters of a chess games, including the openings [6], [7], [8], [9], [10].

This paper has following structure. After Introduction, we will define Opening Tablebases and present basic algorithm. In the next section we will show the original application and procedure for incremental machine learning.

2. Opening Tablebases - Definition

Exploring this field, the author has come to significant results in the definition, application and implementation of a special

type of base opening, which are directly coupled and involved in the program system Axon [11], [16], [17], [22]. The first step was to define an efficient base model with the possibility of the fast access to individual items in the opening table [12], [13]. The author has designed a new type of base opening named *OpenX*, which in addition to the basic position of the table contains an index and a main table that significantly accelerates access to some areas of base [22].

The idea could be pointed in this way:

• Having at disposal direct control over all resources of transposition tables, it was possible to develop a specific type of procedure that allow machine learning, thus the automatic synthesis of new knowledge in transposition table without the use of expert knowledge or human influence for setup of knowledge [14] .

This principle represents a significant advance because the construction of new knowledge in the table opening is very efficient, especially in varieties of machine learning from PGN game against other computers.

2.1 The Basic Algorithm

However, we will explain the format of the tables showing the types and data structures, using the programming language Pascal [15], [16]:

```
type
      oaxmitype = record {Base data type}
            move:word; {Move definition}
      value:word; {Move value}
end;
oaxtype = record
      eco:string[3]; { ECO code of opening}
      {.}
on_move:Boolean; {Move side}
manual:Boolean; {Manuel tuning}
      {.}
      position:array [0..31] of byte; {Position definition}
      omi:array [1..10] of oaxmitype; {Moves and Weights}
      {.}
version:longint; {Type;Program version}
depth:byte; {Depth}
move:integer; {Best move}
evaluation:integer; {Searcher evaluation}
{.}
      name:string[64]; {Variant name}
      next:longint; {Pointer to the next position}
end;
```

The basic structure of the data that are defined and different types of use are:

```
var
OAX,DAX:oaxtype; {Working and support position}
index:longint; {Index}
```

```
openbase:Boolean; {Base is active}
oaxidx:file of longint; {Index file}
oaxdat,maxdat,ecodat: file of oaxtype;{Main base definition}
opendir : string[32]; {Working base directory, mainly C:/axon/}
```

3. Basic Procedures and Applications

The basic definition of the base contains 232516 positions from all varieties of opening. Positions are located in primary OAX.DAT file which has 354 Kb and index file capacity of 16,384 Kb. The following figure shows the visual appearance of the application OpenX, which is a database server for all types of Axon programs which have a standard interface to the database OAX:

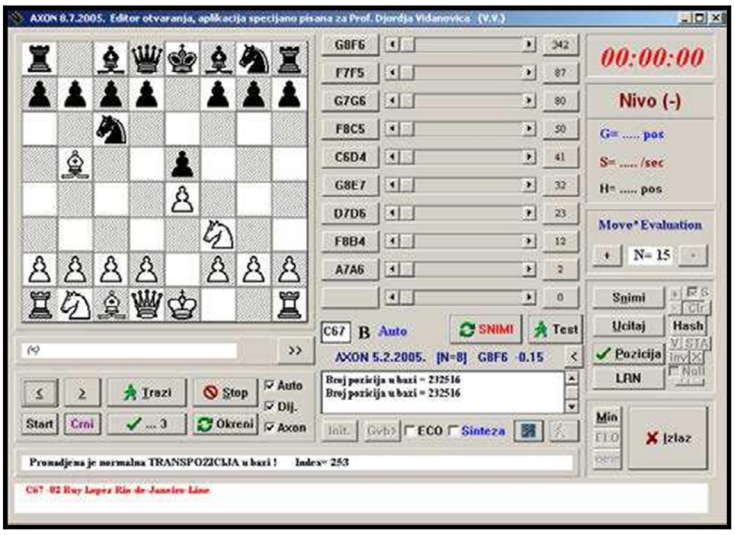

Figure 1. The visual appearance of the application which is the OpenX server for the Axon software

The application OpenX has all basic and a variety of advanced features (Figure 1). For instance, it is possible to automatically and manually tune all move weights in every trasposited position. Also, every position could be analyzed with build-in chess engine Axon. This special feature enables that the opening base in perfectly manually tuned to the engine.

Next figure shows a portion of OpenX application that implements features that are related to machine learning (Figure 2):

3.1 The procedure of Machine Learning from PGN

Scenario of machine learning can be shown in the following steps:

• The first phase is the implementation of automatic playing the game using a standard graphical environment [18], [19]. Played games and along with the results are stored internally in the level GUI.

• By using of OpenX applications games which played game are analyzed at the level of PGN files [20], [21]. To organize approach to automatic generator, program first converts PGN files to middle-file that contains all the essential information relevant for automated synthesis [23], [24]. For example, if the input has PGN file of the following form:

```
[Event "Arena tournament"] {No=1}
[Site "PC15"]
[Date "2006.02.21"]
[Round "1"]
```

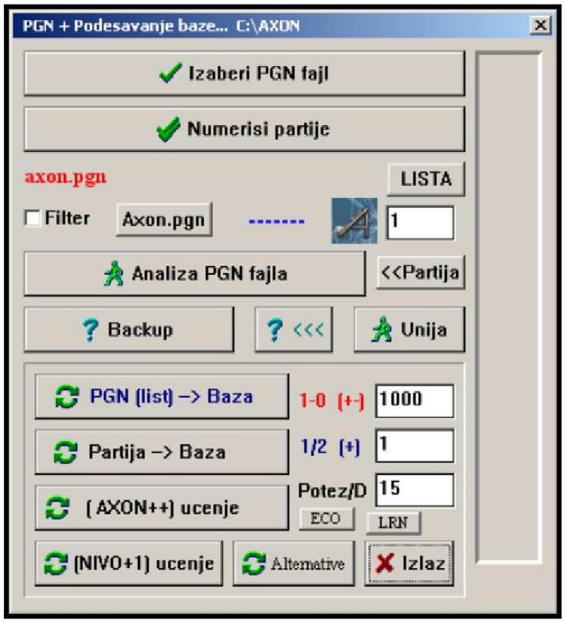

Figure 2. The figure shows the options of machine learning

[White "Axon"]

[Black "Fruit_21"]

[Result "1-0"]

[ECO "C42"]

[Opening "Petrov"]

[Time "20:32:38"]

[Variation "Damiano Variation"]

[TimeControl "40/600:40/600:40/600"]

[Termination "adjudication"]

[PlyCount "117"]

[WhiteType "program"]

[BlackType "program"]

1.e4 e5 {(e7e5 Nb1c3 Ng8f6 Ng1f3 Nb8c6 Bf1b5 Bf8b4 d2d3 00 Bb5xc6 d7xc6 Nf3xe5 Qd8d4 Ne5c4 Bb4xc3+ b2xc3 Qd4xc3+ Bc1d2) -0.15/13 33} 2.Nf3 Nf6 {(Ng8f6 Nb1c3 Nb8c6 Bf1c4 Bf8c5 00 00 d2d3 d7d6 Bc1g5 Bc8e6 Nc3d5) -0.16/12 15} 3.Nxe5 Qe7 {(Qd8e7 Ne5f3 Qe7xe4+ Bf1e2 Nb8c6 Nb1c3 Qe4b4 Nc3b5 Qb4a5 00 a7a6 Nb5d4 Nf6e4) -0.38/12 23} 4.Nf3 {(Ne5f3) 0.30/14 29} Qxe4+ {(Qe7xe4+ Bf1e2 Bf8e7 Nb1c3 Qe4f5 00 00 d2d4 Nb8c6 Be2d3 Qf5g4 h2h3 Qg4e6) -0.49/12 24} 5.Be2 {(Bf1e2) 0.40/16 1} Be7 {(Bf8e7 d2d4 00 Nb1c3 Qe4f5 00 Nb8c6 Be2d3 Qf5h5 Bc1f4 Nf6d5 Nc3xd5 Qh5xd5) - 0.62/12 19} 6.Nc3 {(Nb1c3) 0.40/14 12} Qb4 {(Qe4b4 00 00 d2d4 d7d5 Nc3b5 Qb4a5 Bc1d2 Qa5b6 Rf1e1 Nb8c6 Be2d3) -0.64/13 15} 6.O-O {(00) 0.40/14 15} d5 {(d7d5 d2d4 00 Nc3b5 Qb4a5 Qd1e1 Qa5b6 Be2d3

```
Nb8c6 Bc1f4 Nf6e4 Nb5xc7) -0.70/13 14} 8.d4 {(d2d4) 0.40/13 10} O-O {(00 Nc3b5 Qb4a5
Bc1d2 Qa5b6 a2a4 a7a5 Bd2f4 Nb8a6 Rf1e1 Nf6e4 Nf3e5) -0.60/12 23} 9.Re1 {(Rf1e1)
0.40/14 15} Qb6 {(Qb4b6 Be2b5 Be7b4 Qd1d3 Bc8d7 Bb5xd7 Nb8xd7 Bc1f4 Rf8e8 Nf3g5 Bb4xc3
b2xc3) -0.42/11 13} 10.Ne5 {(Nf3e5) 0.40/13 15} Re8 {(Rf8e8 Be2d3 Be7d6 Nc3b5 Bd6xe5
Re1xe5 Re8xe5 d4xe5 Nf6g4 Qd1f3 Nb8c6 Bc1f4 Nc6xe5 Bf4xe5 Ng4xe5 Qf3xd5 Ne5xd3 Qd5xd3)
```

The converted file will use the following structure:

```
E2E4 <=> P ** E4        (e4 , Axon=W) (1)
E7E5 <=> P ** E5        (e5 , Axon=W) (2)
G1F3 <=> N ** F3        (Nf3 , Axon=W) (3)
G8F6 <=> N ** F6        (Nf6 , Axon=W) (4)
F3E5 <=> N ** E5        (Nxe5 , Axon=W) (5)
D8E7 <=> Q ** E7        (Qe7 , Axon=W) (6)
E5F3 <=> N ** F3        (Nf3, Axon=W) (7)
E7E4 <=> Q ** E4        (Qxe4+ , Axon=W) (8)
F1E2 <=> B ** E2        (Be2 , Axon=W) (9)
F8E7 <=> B ** E7        (Be7 , Axon=W) (10)
B1C3 <=> N ** C3        (Nc3 , Axon=W) (11)
E4B4 <=> Q ** B4        (Qb4 , Axon=W) (12)
E1G1 <=> K E1 G1        (O-O , Axon=W) (13)
D7D5 <=> P ** D5        (d5 , Axon=W) (14)
D2D4 <=> P ** D4        (d4 , Axon=W) (15)
E8G8 <=> K E8 G8        (O-O , Axon=W) (16)
F1E1 <=> R ** E1        (Re1 , Axon=W) (17)
B4B6 <=> Q ** B6        (Qb6 , Axon=W) (18)
F3E5 <=> N ** E5        (Ne5 , Axon=W) (19)
F8E8 <=> R ** E8        (Re8 , Axon=W) (20)
E2D3 <=> B ** D3        (Bd3 , Axon=W) (21)
B8D7 <=> N B* D7        (Nbd7 , Axon=W) (22)
C3A4 <=> N ** A4        (Na4 , Axon=W) (23)
```

Procedure that transforms PGN to this middle-code is in fact the lexical analyzer of PGN files. It is the integral part of OpenX procedure.

3.2 Realization of the Procedure of Machine Learning

In the third stage, based on the compressed list that is displayed, the algorithm performs the setting of the trees in the transposition base opening [22], [25]. The moves in the variants that Axon does not prefer, and those who lead to the lost game, are become weaker. In the case that the current depth of the opening in the base is not sufficient, program performs the synthesis of new nodes. In this way, the system itself works automatically, using the basic knowledge of the game against an opponent, learns and adjusts its own base, adapting it to the game of opponents [22], [26]. If in the second iteration the position repeats, now with knowledge of the position of learned database, it could played better. The results of ELO rating increases by 50-70 ELO points. The described process can be represented by the following diagram (Figure 3.):

Therefore, the principle of the creation is fully automated, synthetic opening base becomes a new useful method which

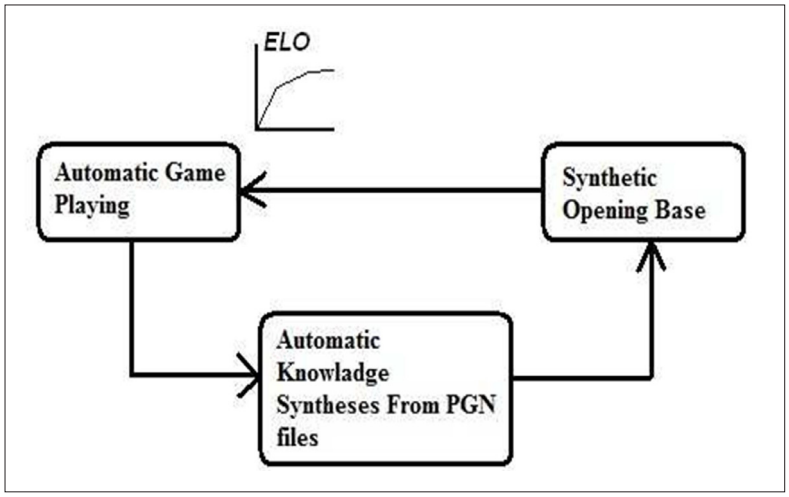

Figure 3. The principle of automatic reinforcement strength program using a synthetic base opening system

introduces the idea of automated learning in a very useful way in the field of computer chess.

4. Conclusion

The basic method of new approach in machine learning in the area of synthetic opening base is presented in this paper [22]. We concentrated on novel idea and presented requested forms and methods for its realization. Apart from represented basic method, OPENX program contains a few more methods of machine learning which are less important. One of the methods of incremental of the depth of evaluation of each node in the base.

By using of incorporated program AXON I the system is able to automatically verify the evaluation of each node in the base by increasing the depth of verification for 1 in each phase. The system performs this process by analysing of transpositional base of opening. The the importance of this as well as other systems of automatical learning should be verified particular. The principle of synthetic base of opening was tested and very successfully applied in practical and tournament practice.

Acknowledgement

This paper is supported by the interdisciplinary project III44006 of the Ministry of Science and Technology of Republic of Serbia.

References

[1] Buro, M. (1999). Toward Opening Book Learning. *ICCA Journal*, Vol. 22, No. 2, pp. 98-102. ISSN 0920-234X.

[2] Slate, D. J. (1987). A Chess Program that uses its Transposition Table to Learn from Experience. *ICCA Journal,* 10 (2) 59-71. ISSN 0920-234X.

[3] Buro, M. (2000). Toward Opening Book Learning. *Games in AI Research* (eds. H.J. van den Herik and H. IIda), p. 47-54. Universiteit Maastricht, Maastricht, The Netherlands. ISBN 90-621-6416-1.

[4] Hyatt, R. M. (1999). Book Learning - a Methodology to Tune an Opening Book Automatically. *ICCA Journal*, 22 (1) 3-12.

[5] Spracklen, K. (1983). Tutorial: Representation of an Opening Tree. *ICCA Newsletter,* 6 (1) 6-8.

[6] Schaeffer, J., Hlynka, M., Jussila, V. (2001). Temporal Difference Learning Applied to a High-Performance Game-Playing Program. *Proceedings International Joint Conference on Artificial Intelligence*, p. 529-534.

[7] Baxter, J., Tridgell, A., Weaver, L. (1998). Experiments in Parameter Learning using Temporal Differences. *ICCA*

Journal, 21 (2) 84-99.

[8] Beal, D. F., Smith, M. C. (1999). Learning Piece-Square Values using Temporal Differences. *ICCA Journal*, 22 (4) 223-235.

[9] Tesauro, G. (1992). Practical Issues in Temporal Difference Learning. *Machine Learning,* Vol. 8, p. 257-277. ISSN 0885-6125.

[10] Adelson-Velsky, G., Arlazarov, V., Bitman, A., Zhivotovsky, A., Uskov, A. (1969). Programming a computer to play chess, *In*: Proceedings of the 1st Summer School on Mathematical Programming, Vol. 2, (1969) p. 216-252.

[11] Donninger, Chr.(1993). Null Move and Deep Search: Selective- Search Heuristics for Obtuse Chess Programs, *ICCA Journal*, 16 (3) 137-143. ISSN 0920-234X.

[12] Shannon, C. E. (1988). Programming a Computer for Playing Chess, *Philosophical Magazine*, 41 (7) 256-275. Reprinted (1988) in Computer Games I (ed. D.N.L. Levy), p. 81-88. Springer-Verlag, New York, N.Y. (1950) ISBN 0-387-96496-7.

[13] Slate, D. J., Atkin, L. R. (1983). CHESS 4.5 – The Northwestern University Chess Program, *Chess Skill in Man and Machine* (ed. P.W. Frey), p. 82-118. Springer-Verlag, New York, N.Y. 2nd ed. 1983. (1977) ISBN 0-387-90815-3.

[14] Vuckovic, V., Vidanovic, D. (2007). An algorithm for the Detection of Move Repetition Without the use of Hash-Keys, *Yugoslav Journal of Operations Research (YUJOR*), 17 (2) 257- 274. Belgrade, Serbia. ISSN 0354-0243.

[15] Vuckovic, V. (2006). The Theoretical and Practical Application of the Advanced Chess Algorithms, *PhD Theses*, The Faculty of Electronic Engineering, The University of Nis, Serbia (2006).

[16] Vuckovic, V. (2001). *Axon/Achilles* experimental chess engines information could be find at: http://axon.elfak.ni.ac.rs, http://chess.elfak.ni.ac.rs

[17] Zobrist, A. L. (1970). *A New Hashing Method with Application for Game Playing*. Technical Report #88, Computer Science Department, The University of Wisconsin, Madison, WI, USA. Reprinted (1990) in *ICCA Journal*, 13 (2) 69-73. ISSN 0920-234X.

[18] Vuckovic, V. (2008). The Compact Chessboard Representation, *ICGA Journal*, Tilburg, The Netherlands, (September). ISSN 1389-6911. 31 (3) 157 - 164.

[19] Vuckovic, V. (2007). The Method of the Chess Search Algorithms Parallelization using Two-Processor Distributed System", The Scientific Journal Facta Universitatis, Series Mathematics and Informatics, Niš 2007, ISSN 0352- 9665. 22 (2) 175-188.

[20] Solak, R., Vuckovic, V. (2010). Time Management During a Chess Game, *ICGA Journal*, Tilburg, The Netherlands, 2010, ISSN 1389-6911. 32 (4) 206- 220.

[21] Vuckovic, V. (2011). Napredni šahovski algoritmi i sistemi, monografija, Zaduzbina Andrejevic, Biblioteka Dissertatio, ISSN 0354-7671, Beograd.

[22] Vuckovic, V. (2007). The Realization of the Parallel Chess System Using UDP Communication Protocol, *In: Proceedings of the* VIII International Conference on Telecommunications in Modern *Satellite, Cable and Broadcasting Services* TELSIKS, Niš, 26-28. (September), Volume 2, p. 450-453. (IEEE Catalog Number: 07EX1875) (ISBN 1-4244-1467-9)

[23] В. Вучковић, "Реализација ефикасног генератора потеза у шаховским апликацијама високих перформанси ", *Зборник радова са 52. Конференције* ЕТРАН-а CD ROM Proceedings, Секција Вештачка интелигенција, рад VI2.3, Палић, 8-12. јуни 2008, (ISBN 978-86-80509-63-1).

[24] П. Рајковић, В. Вучковић, "Основни елементи хеуристичке евалуационе функције", *Зборник радова са 52. Конференције* ЕТРАН-а, CD ROM Proceedings, Секција Вештачка интелигенција, рад VI2.4, Палић, 8-12. јуни 2008, (ISBN 978-86-80509-63-1).

[25] В. Вучковић, "Специјални елементи евалуационе функције", Зборник радова са 53. Конференције ЕТРАН-а, CD ROM Proceedings, Секција Вештачка интелигенција, рад VI1.3, Vrnjačka Banja, 15 – 18. juna 2009. (ISBN 978-86-80509-64-8)