A Principled Method for Exploiting Opening Books

Romaric Gaudel, Jean-Baptiste Hoock, Julien Pérez, Nataliya Sokolovska, and Olivier Teytaud

LRI, CNRS UMR 8623 & INRIA-Saclay,
bât. 490, Univ. Paris-Sud, F-91405 Orsay Cedex, France
`firstname.name@lri.fr`

Abstract. In the past we used a great deal of computational power and human expertise for storing a rather big dataset of good 9x9 Go games, in order to build an opening book. We improved the algorithm used for generating and storing these games considerably. However, the results were not very robust, as (i) opening books are definitely not transitive, making the non-regression testing extremely difficult, (ii) different time settings lead to opposite conclusions, because a good opening for a game with 10s per move on a single core is quite different from a good opening for a game with 30s per move on a 32-cores machine, and (iii) some very bad moves sometimes still occur. In this paper, we formalize the optimization of an opening book as a matrix game, compute the Nash equilibrium, and conclude that a naturally randomized opening book provides optimal performance (in the sense of Nash equilibria). Moreover, our research showed that from a finite set of opening books, we can choose a distribution on these opening books so that the resultant randomly constructed opening book has a significantly better performance than each of the deterministic opening books.

1 Introduction

It is widely known that opening books (OB) are crucial in many games [4, 11, 9, 12]. Incidentally, it can be crucial also for other applications: carefully choosing the first decision in a planning problem is often quite important. For example, in power plant management, taking care of strategical decisions (in particular, the quantity of hydroelectric stocks preserved) before the winter is rather important a long time before the moment at which the situation is visibly critical.

An adequate representation of the opening book is not so easy. Usually, opening books are built from sets of games (see, e.g., [11]), and possibly (for games in which alphabeta works) extended by alpha-beta techniques, typically with iterative deepening or related methods [7, 9]. Then, several solutions are as follows:

- **Rote-learning.** Just keep all the archive, and when a sequence of moves is equal to the prefix of a game g in the archive, play the same move as in g. If several moves are available, select the one which most frequently leads to a win. Do not follow a move with success rate below a threshold; [11] suggests to avoid moves below 50 %. There are implementations in which moves played less than a fixed number of times are canceled; this reminds the classical confidence/support thresholds when rules were extracted from databases.

- Q**-functions.** An improvement consists in replacing the archive of games by a set of pairs (situation, move). This is quite reasonable as in some games, permutations of games are also good games, and if a move is good in a situation, it is good whatever may be the sequence of moves leading to this situation.
- V**-functions.** We might want to generalize the permutation approach above: instead of keeping pairs, just keep situations, and play a move if it leads to a situation with a success rate above 50 %. This is advocated in, e.g., [2], and empirically strongly increases the average number of moves in the opening book.

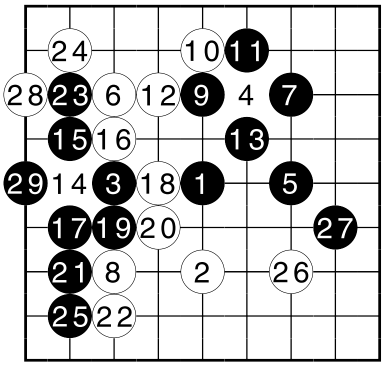

Fig. 1. Here MoGoTW was White. The human, a quite strong professional player but with little experience in 9x9, was in very bad situation at move 7 - at this point it was easy for White to keep two groups alive, one around stones 2 and 8, and one around stones 4 and 6. But MoGoTW did not reply to move 7, and played the useless move 8 which strengthens the group at the bottom whereas it was safe, and did not protect the group in the top which was under attack. MoGoTW was just selecting moves leading to situations which were in its opening book! This shows that representing an opening book by $\hat{V}$-functions (i.e., functions mapping values to situations) can lead to big troubles - when the situation is rare due to mistakes by the opponent, $\hat{V}$ is known only for moves which equilibrate the situation and therefore using the $\hat{V}$ function might destroy the advantage. This leads to extremely long sequences in the opening book, but it would be much better to exit the opening book earlier (here after move 7). Interestingly, MoGoTW played all the stones in this figure in the opening - showing the success of the approach based on $\hat{V}$ from the point of view of the length of the opening sequence - but not in terms of quality of moves.

As stated above, building an opening book is far from an easy task. For games in which alpha-beta is efficient, there are natural tools for this task (typically, iterative deepening [9]). But for difficult games, such as Go, in which only Monte-Carlo Tree Search provides good results, there is not a large number of tools yet. [2] proposed algorithms for the building of opening books, including experiments on grids, with good results against the baseline; however, when the algorithm is applied against humans, several very bad moves appeared, suggesting that the performance against humans does not follow the performance against computers. A typical example is given in [2] in which it is shown that adding human expertise provides a huge speed-up. A particularly impressive example of bad behavior is the game shown in Fig. 1, in which MoGoTW played some very bad moves and lost against a 8P player, whereas the human had made big mistakes in the early stages of the opening. As pointed out in [5], taking care of avoiding bad moves is indeed much more important than taking care of adding good moves.

We here show that indeed, $\hat{V}$-functions are quite dangerous, in spite of the fact that they strongly increase the length of the opening sequence, and that a simple modification of $\hat{V}$ algorithms make them much more reliable. Section 2 presents our modified algorithms. Section 3 will show how to combine several deterministic strategies into an optimal randomized strategy. Section 4 presents our experimental results. Section 5 concludes by a summary and future work.

Notations

In the paper, $nb\ games(s)$ is the number of games in the archive including situation s, and $nb\ wins(s)$ is the number of won games in the archive including situation s. $parent(s)$ is the parent situation of situation s; in some cases there are several such parent situations, and in this case $parent(s)$ is the situation which was met before situation s in the context. $grandParent(s) = parent(parent(s))$.

2 Simple Modifications for $\hat{V}$-Algorithms

The baseline algorithm (termed "default" in the sequel) is shown in Alg. 1.

The situation we want to avoid is depicted in Fig. 2.

Algorithm 1. The "default" algorithm for opening book. This is the baseline in our experiments

```
Goal: Select a move by opening from the situation $s$.
Define: $\hat{V}(situation) \leftarrow$ percentage of won games in archive containing $situation$.
Define: $transition(s, s')$ means that playing a move can lead to $s'$ from $s$.
if There is $s'$ such that $transition(s, s')$ and $\hat{V}(s') \geq 0.5$ then
    Return $\arg\max_{s';transition(s,s')} \hat{V}(s')$.
end if
```

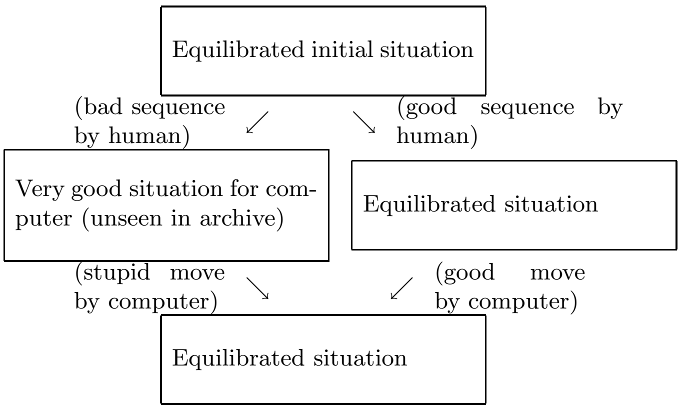

Fig. 2. A bad case we want to avoid. The arrow "stupid move by computer" is the move that is selected in the baseline and that we want to remove. The situation that "attracts" the opening book is not necessarily very good in the default algorithm: it is just the statistically best situation already stored in the archive that can be reached from this situation, and this situation is not necessarily good.

A first natural modification is to choose moves with good statistical guarantees, taking into account the sample size by confidence bounds. This is proposed in the Lower Confidence Bound algorithm (Alg. 2).

Algorithm 2. The "Lower Confidence Bound" (LCB) version of the algorithm.

```
Goal: Select a move by opening from the situation $s$.
Define: $\hat{V}(situation) \leftarrow$ percentage of won games in archive containing $situation$, minus $2/\sqrt{nb\ games(situation)}$.
Define: $transition(s, s')$ means that playing a move can lead to $s'$ from $s$.
if There is $s'$ such that $transition(s, s')$ and $\hat{V}(s') \geq 0.5$ then
    Return $\arg\max_{s'; transition(s,s')} \hat{V}(s')$.
end if
```

We will see also another simple modification: if you leave the opening book, then never consider it again. This is proposed in Alg. 3.

The fourth proposed modification consists in accepting an opening move only if the success rate is higher than the previously seen success rate. This is detailed in Alg. 4.

Finally, it has been suggested in [8, 3] to use a regularized form of the winning rate; we propose this in the "regularized" algorithm[1] (Alg. 5).

Please note that this is not equivalent to coming back to $\hat{Q}$-representations.

[1] Please note that ranking moves by $(nb\ wins)/(nb\ games)$ or by $(nb\ wins + \frac{1}{2})/(nb\ games + 1)$ are two different things (consider, e.g., a move with $nb\ wins = 2$, $nb\ games = 3$ and a move with $nb\ wins = 19,999$, $nb\ games = 30,000$; the first move is preferred in the first case but not in the second).

Algorithm 3. The "non-reentrant" (NR) version of the algorithm.

```
Goal: Select a move by opening from the situation $s$.
if $s$ is not in the archive then
    Return no move
end if
Define: $\hat{V}(situation) \leftarrow$ percentage of won games in archive containing $situation$.
Define: $transition(s, s')$ means that playing a move can lead to $s'$ from $s$.
if There is $s'$ such that $transition(s, s')$ and $\hat{V}(s') \geq 0.5$ then
    Return $\arg\max_{s';transition(s,s')} \hat{V}(s')$.
end if
```

Algorithm 4. The "progress" algorithm for opening book.

```
Goal: Select a move by opening from the situation $s$.
Define: $\hat{V}(situation) \leftarrow$ percentage of won games in archive containing $situation$.
Define: $s'' = grandParent(s)$.
Define: $transition(s, s')$ means that playing a move can lead to $s'$ from $s$.
if There is $s'$ such that $transition(s, s')$ and $\hat{V}(s') \geq 0.5$ and $\hat{V}(s') > \hat{V}(s'')$ then
    Return $\arg\max_{s';transition(s,s')} \hat{V}(s')$.
end if
```

Algorithm 5. The "regularized" (Reg) algorithm for opening book; this is the so-called "even-game" prior[6]. A second regularized version (Reg2) is considered, using $(nb\ wins+100)/(nb\ games+200)$; this increases the strength of the "even-game" prior.

```
Goal: Select a move by opening from the situation $s$.
Define: $\hat{V}(situation) \leftarrow (nb\ wins(situation) + \frac{1}{2})/(nb\ games(situation) + 1)$.
Define: $s'' = grandParent(s)$.
Define: $transition(s, s')$ means that playing a move can lead to $s'$ from $s$.
if There is $s'$ such that $transition(s, s')$ and $\hat{V}(s') \geq 0.5$ and $\hat{V}(s') > \hat{V}(s'')$ then
    Return $\arg\max_{s';transition(s,s')} \hat{V}(s')$.
end if
```

3 Mixing Deterministic Opening Books: Fictitious Play and Matrix Games

Fictitious play is an algorithm for solving zero-sum matrix games. Consider M a matrix with p rows and q columns; player 1 chooses a row i, player 2 chooses a column j, and player 2 pays $M_{i,j}$ to player 1; this means that the reward for player 1 is $M_{i,j}$ and the reward for player 2 is $-M_{i,j}$.

Pure strategies. A pure strategy for player 1 (resp. player 2) is a deterministic policy for playing the game: it is the index of a row (resp. the index of a column).

Mixed strategy. A mixed strategy for player 1 (resp. player 2) is a distribution of probability on pure strategies of player 1 (resp. player 2). The **support** of a mixed strategy is the number of pure strategies with non-zero probability in it.

A **Nash equilibrium** of the game is a pair $(x, y) \in \mathbb{R}^p \times \mathbb{R}^q$ with $x, y \geq 0$ and $||x||_1 = ||y||_1 = 1$ such that

$$\forall y' \in \mathbb{R}^q, y' \geq 0, ||y'||_1 = 1 : x^t M y \leq x^t M y';$$
$$\forall x' \in \mathbb{R}^p, x' \geq 0, ||x'||_1 = 1 : x^t M y \geq {x'}^t M y.$$

A **best response** for player 1 (resp. 2) to a strategy s of player 2 (resp. 1) is a pure strategy which maximizes the expected reward against s. s is not necessarily a pure strategy, but a best response is a pure strategy.

Define $e_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^p$, $e_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^p$, $\ldots$, $e_p = (0, 0, \ldots, 0, 1) \in \mathbb{R}^p$. Define ${e'}_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^q$, ${e'}_2 = (0, 1, 0, \ldots, 0) \in \mathbb{R}^q$, $\ldots$, ${e'}_q = (0, 0, \ldots, 0, 1) \in \mathbb{R}^q$. Consider $z_i = (x_i, y_i)$ for $i \in \mathbb{N}$, with $x_i \in [0, 1]^p$, $y_i \in [0, 1]^q$. z is a **fictitious play** for M if

- $||x_1||_1 = 1$, $||y_1||_1 = 1$;
- For all i even, $y_{i+1} = y_i$ and $x_{i+1} = \frac{ix_i + e_{r_i}}{i+1}$ where r_i is a best response to y_i.
- For all i odd, $x_{i+1} = x_i$ and $y_{i+1} = \frac{iy_i + {e'}_{r_i}}{i+1}$ where r_i is a best response to x_i.

Fictitious play is known to converge since [10], in the sense that for all zero-sum games, its accumulation contains only Nash equilibria.

4 Experiments

Opening books are not transitive; one can have an opening book that is quite strong against a given opponent, and not against another. Also, the comparison between two opening books might depend on the considered hardware and the computational power. How to build an opening book, from various opening books with no clear ranking? We propose an answer based on matrix games. The following three elements are crucial.

- **Randomized opening books.** It is known that even when there is a deterministic perfect player (what is the case for Go), it is often much better (from a complexity perspective) to use a randomized solution. Therefore, randomized opening books should be considered. This is often done in order to introduce diversity in games, but the new thing in our work is that we will do this for improved performance, even against opponents with no memory of the past or met only once, and not (only) for diversity and for the pleasure of human opponents.
- **Matrix solving.** Given strategies numbered $1, 2, \ldots, K$ for playing the opening with a book of games, we can build the matrix M where $M_{i,j}$ is the success rate of strategy i as Black against strategy j as White. This

is a matrix game. When there is no transitivity, as in the case of opening books, optimal strategies may be mixed strategies (i.e., randomized): they consist in a distribution of probability on pure (i.e., deterministic) strategies. Whenever optimal players in Go can (provably) be deterministic, the optimal combination of opening book strategies is randomized.
- **Automatization.** It is known (and recalled in experiments above) that opening books provide plenty of surprises such as non-transitivity and dependency on the precise conditions of the game. Therefore, all the processes of choosing the opening strategy should be made automatically, for all game conditions.

We propose the following three-step solution.

- (1) Build the matrix M discussed above.
- (2) Apply fictitious play for solving this matrix game (see section 3); this provides a distribution of probability p_b for Black and a distribution of probability p_w for White.
- (3) The resulting stochastic opening book is then as follows:
  - As Black, play strategy i with probability $p_b(i)$.
  - As White, play strategy i with probability $p_w(i)$.

We tested our algorithms and arrived at the results presented in Table 1.

Comparing the various deterministic strategies is difficult: in some cases, Reg2 (in which statistics are rather regularized) is quite bad (e.g., for White), but in other cases it performs rather well (e.g., for Black with 1,500 simulations per move). Also, the use of lower bounds (LCB) is sometimes quite successful (in particular for Black at 15,000 simulations per move). This is somehow related to [1]: for difficult problems, it might be better to optimize the parameter of a strategy (in particular a randomized strategy) than applying sophisticated tools for approximating value functions.

For the use of mixed strategies, the results can be summarized as follows.

- With 15,000 simulations per move, the mixed strategy for Black reaches success rate 28.0 % at least against all opponents; whereas each pure strategy reaches success rate 26.9 % at most. As White, the mixed strategy can reach 72.0 % against any opponent, whereas the pure strategies cannot perform better than 71.5%.
- With 150,000 simulations per move, the best strategy is deterministic - it is "Reg", both for Black and White. Reg2 was not included in the test.
- With 1,500 simulations per move, the best strategy is deterministic - it is "Reg" for black, and LCB for white.

We emphasize two different advantages of randomized strategies.

- First, in some cases, the mixed strategies are better; however, the advantage is minor in our experiments.
- Second, fictitious play provides a principled tool for optimizing mixed strategies; it is anytime (it is an iterative method), simple and proved.

Table 1. Scores of the various opening books as Black against the various opening books as White; following the tradition in game theory, the success rates are those of the "row" player (i.e., Black). The probabilities are the Nash equilibrium: the optimal strategy is deterministic except for 15,000 simulations per move. The Nash equilibrium is found by fictitious play, with 10,000 iterations; each number in matrices above is found by averaging 5,000 games.

Black \ White	Default	LCB	LCB +NR	NR	Reg (NR)	Reg2 (NR)	Probas for Black
1,500 simulations per move							
Default	29.3	28.8	55.5	55.6	25.9	57.7	0
LCB	28.9	26.8	54.5	52.8	28.5	50.8	0
LCB+NR	28.2	30.2	53.6	53.8	23.5	52.5	0
NR	30.7	30.8	56.4	54.8	26.9	55.7	0
Reg	25.8	31.9	55.7	53	23.8	52.5	0
Reg2	36.1	36.1	50.8	44.3	37.7	59	100
Probas for White	0	100	0	0	0	0	
15,000 simulations per move							
Default	29.3	28.8	55.5	55.6	25.9	55.7	0
LCB	28.9	26.8	54.5	52.8	28.5	50.8	69.6
LCB+NR	28.2	30.2	53.6	53.8	23.5	52.5	0
NR	30.7	30.8	56.4	54.8	26.9	51.6	30.4
Reg	25.8	31.9	55.7	53	23.8	67.7	0
Reg2	35.5	22.6	48.4	51.6	22.6	61.3	0
Probas for White	0	28.6	0	0	71.4	0	
150,000 simulations per move							
Default	29.3	28.8	55.5	55.6	21.2	NA	0
LCB	28.9	26.8	54.5	52.8	19.2	NA	0
LCB+NR	28.2	30.2	53.6	53.8	15.4	NA	0
NR	30.7	30.8	56.4	54.8	18.5	NA	0
Reg	24.6	26.2	48.8	56.2	24	NA	100
Probas for White	0	0	0	0	100		

It is interesting to consider what humans would choose if trying to choose between the pure (deterministic) strategies from the tables of results. It is likely that a human would choose, for 150,000 simulations per move, NR for Black, whereas it performs extremely bad against some white opponents; the principled solution (REG) reaches 24% whereas NR reaches 18.5%.

5 Conclusion

First, we discussed the various techniques for constructing an opening book from a finite set of games. We clearly see that some details, such as our "NR" modification or the regularization, have a huge unsuspected impact on the results; also, the results of a given technique are quite different depending on the considered problem; just changing the number of simulations, or, more importantly, considering White or Black, makes "NR" or "LCB" or regularization quite good or rather bad. This suggests the use of an automatic choice between several techniques. LCB and NR usually perform well for White, and for Black the situation highly depends on the number of simulations. An immediate further work consists in adding some other parameters; after all, this looks like a direct policy search applied to the building of opening book, and this might be the best approach for complex problems. A related work emphasizing such an approach (including randomization) is [1], in a different framework.

Second, we considered fictitious play as a tool for simultaneously randomizing and optimizing opening books. The resulting procedure can be automatized for a given technical setting (time per move, hardware), and provides results that would not be guessed by handcrafting. The results are provably Nash equilibria.

References

[1] Amato, C., Bernstein, D., Zilberstein, S.: Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. In: AAMAS (2009)

[2] Audouard, P., Chaslot, G., Hoock, J.-B., Perez, J., Rimmel, A., Teytaud, O.: Grid coevolution for adaptive simulations: Application to the building of opening books in the game of go. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 323–332. Springer, Heidelberg (2009)

[3] Berthier, V., Doghmen, H., Teytaud, O.: Consistency modifications for automatically tuned monte-carlo tree search. In: Blum, C., Battiti, R. (eds.) LION 4. LNCS, vol. 6073, pp. 111–124. Springer, Heidelberg (2010)

[4] Buro, M.: Toward opening book learning. ICCA Journal 22, 98–102 (1999)

[5] Donninger, C., Lorenz, U.: Innovative opening-book handling. In: ACG, pp. 1–10 (2006)

[6] Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: ICML 2007: Proceedings of the 24th International Conference on Machine Learning, pp. 273–280. ACM Press, New York (2007)

[7] Korf, R.E.: Depth-first iterative-deepening: an optimal admissible tree search. Artif. Intell. 27(1), 97–109 (1985)

[8] Lee, C.-S., Wang, M.-H., Chaslot, G., Hoock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C., Hong, T.-P.: The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments. IEEE Transactions on Computational Intelligence and AI in games (2009)

[9] Nagashima, J., Hashimoto, T., Iida, H.: Self-playing-based opening book tuning. New Mathematics and Natural Computation (NMNC) 02(02), 183–194 (2006)

[10] Robinson, J.: An iterative method for solving a game. Annals of Mathematics 54, 296–301 (1951)

[11] Tay, A.: A Beginner's Guide to Building a Opening Book, HorizonChess FAQ (2001)

[12] Walczak, S.: Improving opening book performance through modeling of chess opponents. In: CSC 1996: Proceedings of the 1996 ACM 24th Annual Conference on Computer Science, pp. 53–57. ACM, New York (1996)