The rectangular seeds of Domineering, Atari-Go and Breakthrough

Tristan Cazenave*, Jialin Liu†, Olivier Teytaud†
* LAMSADE, Université Paris-Dauphine, Paris, France
† TAO, Inria, Univ. Paris-Sud, UMR CNRS 8623, France

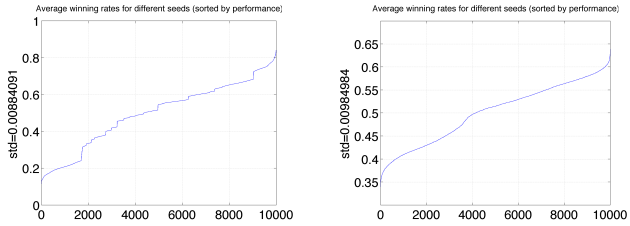
***Abstract*—Recently, a methodology has been proposed for boosting the computational intelligence of randomized game-playing programs. We modify this methodology by working on rectangular, rather than square, matrices; and we apply it to the Domineering game. At CIG 2015, We propose a demo in the case of Go. Hence, players on site can contribute to the scientific validation by playing (in a double blind manner) against both the original algorithm and its boosted version.**

I. Algorithms

Given a stochastic AI, we can check its performance against a baseline program (possibly itself) as we vary the random seed. I.e. we can generate K different random seeds, and for each of these seeds play K_t games against the baseline. We can then plot the winning rates, sort, and compare the variations to the standard deviations. Results are presented in Fig. 1 and show for several games that the seed has a significant impact. The methodologies presented in this paper are based on this phenomenon.

Several works were dedicated to combining several AIs in the past. [2] combines several different AIs. [1] uses Nash methods for combining several opening books. [3] proposed to construct several AIs from a single stochastic one and to combine them by the BestSeed and Nash methods detailed below.

This section presents an overview of two methods proposed in [3] for building a boosted algorithm, from a set of seeds: the Nash-approach and the BestSeed-approach. We also define, for comparison, the uniform portfolio, which is just a uniform sampling of the considered random seeds, as detailed later; it is not stronger than the original AI.



Domineering 5x5 Domineering 9x9

Fig. 1: Impact of the seed on the success rate. For the n^{th} value, we consider the n^{th} worst seed for Black and the n^{th} seed for White, and display their average scores against all opponent seeds. The label on the y-axis shows the standard deviation of these averages; we see that there are good seeds (far above the 50 % success rate, by approx. 12x the standard deviation).

A. Creating a probability distribution on random seeds

Typically, a stochastic computer program uses a random seed. The random seed ω is randomly drawn (using the clock, usually) and then a pseudo-random sequence is generated. Therefore, a stochastic program is in fact a random variable, distributed over deterministic program. AI is our game playing artificial intelligence; it is stochastic. $AI(\omega)$ is a deterministic version; ω is a seed, which is randomly drawn in the original AI. We can easily generate plenty of ω and therefore one stochastic AI becomes several deterministic AIs, termed AI_1, ..., AI_K.

Let us assume then one of the players plays as Black and the other plays as White. We can do the same construction as above for the AI playing as Black and for the AI playing as White. We get AI_1,...,AI_K for Black, and AI'_1, ..., AI'_{K_t} for White. From now on, we present the algorithm for Black - still, for this, we need the AI' as well. The algorithm for enhancing the AI as White is similar. Let us define $M_{i,j} = 1$ when AI_i (playing as Black) wins against AI'_j (playing as White). Otherwise, $M_{i,j} = 0$. Also, let us define $M'_{i,j} = 1$ when AI'_i (playing as White) wins against AI_j (playing as Black) - we have $M'_{i,j} = 1 - M_{j,i}$.

- M (used for the learning for Black) is the matrix of $M_{i,j}$ for $1 \leq i \leq K$ and $1 \leq j \leq K_t$.
- M' (used for the learning for White) is the matrix of $M'_{i,j}$ for $1 \leq i \leq K$ and $1 \leq j \leq K_t$.

If $K_t \leq K$, M and M' have $K_t \times K_t$ entries in common (up to transformation $M'_{i,j} = 1 - M_{j,i}$); therefore building M and M' needs the result of $2K \times K_t - K_t^2$ game results.

Given M, the BestSeed approach consists in selecting one seed. We just pick up i^* such that $\sum_{j=1}^{K_t} M_{i^*,j}$ is maximal. Our BestSeed approach for Black will use random seed i^*; it is a deterministic program. For the Nash approach, we select a probability distribution on seeds. We compute (p, q), a Nash equilibrium of M. The Nash approach for Black will use seed i with probability p_i. The Nash approach provides a stochastic policy, usually stronger than the original policy [3].

We now summarize the two approaches, in Algorithm 1.

Algorithm 1 Approach for boosting a game stochastic game AI.

Input: a stochastic AI playing as Black, a stochastic AI' playing as White.
Output: a boosted AI termed BAI playing as Black, a boosted AI BAI' playing as White.
Build $M_{i,j} = 1$ if AI_i (Black) wins against AI'_j (White) for $i \in \{1, \ldots, K\}$ and $j \in \{1, \ldots, K_t\}$.
Build $M'_{i,j} = 1$ if AI'_i (White) wins against AI_j (Black) for $i \in \{1, \ldots, K\}$ and $j \in \{1, \ldots, K_t\}$.
if BestSeed // deterministic boosted AI **then**
    BAI is AI_i where i maximises $\sum_{j=1}^{K_t} M_{i,j}$.
    BAI' is AI'_i where i maximises $\sum_{j=1}^{K_t} M'_{i,j}$.
end if
if Nash // stochastic boosted AI **then**
    Compute (p, q) a Nash equilibrium of M.
    BAI is AI_i with probability p_i
    Compute (p', q') a Nash equilibrium of M'.
    BAI' is AI'_j with probability p'_i
end if
if Uniform // stochastic AI **then**
    BAI is AI_i with probability $1/K$.
    BAI' is AI'_j with probability $1/K$.
end if

II. Experiments

We provide experiments on the Domineering board game. In all our experiments, we use a MCTS implementation. Domineering is a two-player game with very simple rules: each player in turn puts a tile on empty locations in the board. The game starts with an empty board. The first player who can not play loses the game. Usually, one of the player has vertical 2x1 tiles, and the other has horizontal 1x2 tiles. We consider the following criteria for our boosted AI playing both as Black and White:

- Generate K' seeds, randomly, for Black and K' seeds, randomly, for White.
- Consider the worst success rate SR of our boosted AI playing as White against these K' strategies for Black. Consider the worst success rate SR of our boosted AI playing as Black against these K' strategies for White.
- Our success rate is the average of these two success rates.

This is a strong challenge for K' large; since we consider separately White and Black, we have indeed K'^2 opponent strategies (each of the K' seeds for Black and each of the K' seeds for White) and consider the worst success rate. We will define this opponent as a K'-exploiter.For $K' = 1$, this opponent is playing exactly as the original AI: this is the success rate against a randomly drawn seed. A score $\geq 50\%$ against $K' = 1$ means that we have outperformed the original AI, i.e. boosting has succeeded; but it is satisfactory to have also a better success rate, against $K' > 1$, than the original AI.

In order to validate the method, we take care that our algorithm is tested with a proper cross-validation: the opponent uses seeds which have never been used during the learning of the portfolio. This is done for all our experiments, BestSeed, Uniform, or Nash. For this reason, there is no bias in our

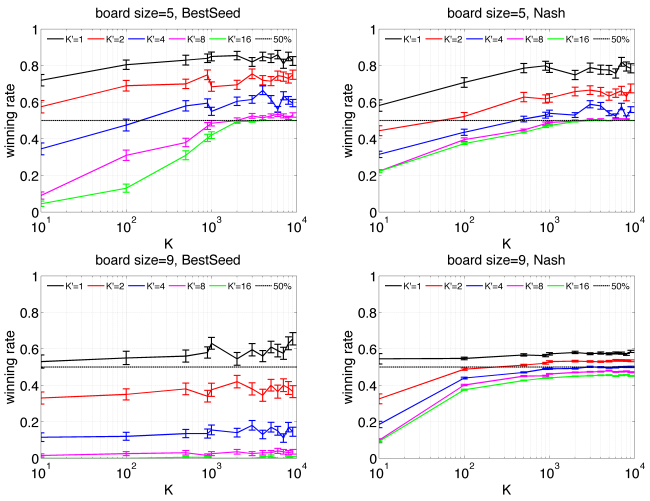


Fig. 2: Results for domineering, with the BestSeed and the Nash approach, against the baseline ($K' = 1$) and the exploiter ($K' > 1$). $K_t = 900$ in all experiments.

results. All results are averaged over 100 runs. Results for Domineering are presented in Fig. 2. In short, BestSeed performs well against the original algorithm (corresponding to $K' = 1$), but its performance against the exploiter ($K' > 1$) is very weak. On the other hand, the Nash version outperforms the original algorithm both in terms of success rate against $K' = 1$ (in all cases) and against $K' > 1$ in most cases (i.e. curves on the middle column in Fig. 2 are better than those on the right column) - however, for breakthrough in large size the results were (very) slightly detrimental for $K' > 1$, i.e. the "exploiter" could learn strategies against it. Our results (winning rate of the boosted algorithm against the non-boosted baseline) are roughly for BestSeed: 86%, 71.5%, 65.5% for Domineering in 5x5, 7x7 and 9x9 respectively. From Fig. 1, we can guess that larger values of K would provide better results. We might see these results as a very simple and effective tool for building an opening book with no development effort, no human expertise, no storage of database. The online computational overhead of the methods used in this paper is negligible, as both for BestSeed and Nash it is just determining the random seed at the beginning of the algorithm. The boosted AIs significantly outperform the baselines. This does not require any source code development.

References

[1] R. Gaudel, J.-B. Hoock, J. Pérez, N. Sokolovska, and O. Teytaud. A Principled Method for Exploiting Opening Books. In *International Conference on Computers and Games*, pages 136–144, Kanazawa, Japon, 2010.

[2] V. Nagarajan, L. S. Marcolino, and M. Tambe. Every team deserves a second chance: Identifying when things go wrong (student abstract version). In *29th Conference on Artificial Intelligence (AAAI 2015), Texas, USA*, 2015.

[3] D. L. Saint-Pierre and O. Teytaud. Nash and the Bandit Approach for Adversarial Portfolios. In *CIG 2014 - Computational Intelligence in Games*, Computational Intelligence in Games, pages 1–7, Dortmund, Germany, Aug. 2014. IEEE, IEEE.