A Simple Tsumego Generator

Ping-Chiang Chou[1], *Shi-Jim Yen[2], Cheng-Wei Chou[2], Ching-Nung Lin[2]
Chang-Shing Lee[3], Olivier Teytaud[4], Hassen Doghmen[4], Tai-Ning Yang[5]

Abstract: Generating problems is an interesting challenge, for both pedagogical purpose and for assessment. We propose a simple solution for generating problems, which can be used on a wide range of games and simulators, and evaluate it in the framework of the game of Go, providing problems from lowest amateur range to problems which are not trivial even for professional players.

Keywords: Go, tsumego, problem generator

1. Problems and Tsumegos

Generating an interesting test case is a classical important problem. A nice test bed is the case of games; but generating problems is also an issue in industry. Most approaches for the game of Go are based on specific generators [3-5]; such tools provide very good Tsumegos, interesting for players of arbitrary strength. Other game-specific tools for generating problems are those based on complexity results, like [7] (generating incredibly complicated ko-fights) and [8] (which provides boards which are not so much bigger than the 19x19 board). However, these tools generate very artificial problems, and are usually too big for the 19x19 board, which makes them not that interesting for humans. Here we focus on methods which can be used for any game. Such a generalist approach has been proposed in [6]; however, the point there was to generate very surprising, unnatural situations; here we want to generate situations:

- possible difficult (we provide a player, and we want the situation to be too hard for him);
- which can occur in real life.

2. A Simple Algorithm: STG

We propose the simple Tsumego generator (STG) as follows, for generating a Tsumego (i.e. an interesting position). The algorithm depends on a strong player (supposed to be strong enough for correctly evaluating positions) and a "weak" player (the position will be too hard for the weak player); it starts in a position which is in favor of the weak player:

Inputs:
1. **a turn-based game, which is unfair, i.e. easier for one side than for the other (e.g. "7x7 Killall-Go with no handicap", or Go with komi 30.5)**
2. **a strong player**
3. **supposed to be practically perfect on this game;**
4. **playing the difficult side.**
5. **a weak player**
6. **playing the easy side;**
7. **supposed to do at least occasionally some mistakes.**

Algorithm:

```
1. found=false
2. While (not found)
3. play one game of the weak player against the strong
   player, until
4. either the strong player loses the game
5. or the strong player detects that the situation is a win;
   then:
6. found ← true.
7. Output the last situation which was a win for the weak
   player
8. Output the move chosen by the strong player on this
   situation
```

Outputs:
1. **a situation**
2. **an optimal move for this situation**

The algorithm is quite simple, but it is efficient, as we will see in next sections.

3. Experiments on the game of Go

We consider two games:

- 7x7 Killall-Go, with no handicap, as in Fig. 1-6. This means that black plays first, then players play as usual, and White wins if and only if he can have at least one

†1 Taiwan Go Association. cabon1224@hotmail.com
†2 Dept. of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan. Email: sjyen@mail.ndhu.edu.tw, kapakapa@gmail.com, jironglin@gmail.com
†3 Dept. of Computer Science and Information Engineering, National University of Tainan, Taiwan. Email: leecs@mail.nutn.edu.tw
†4 TAO team, Inria Saclay IDF, LRI, UMR 8623(CNRS - Universite Paris-Sud), France. Email: olivier.teytaud@gmail.com, hassen.doghmen@gmail.com
†5Department of Computer Science and Information Engineering, Chinese Culture University, Taipei, Taiwan. Email: tnyang@faculty.pccu.edu.tw

stone alive. This is equivalent to 7x7 Go with komi 48.5.

- 7x7 Go with Komi 8.5, as Fig. 7-8. As the fair Komi is supposed to be 9, such a game is supposed to be a win for Black. However, this win for Black is far from being easy.

We define the difficulty of a Tsumego as the rank roughly necessary for (1) finding an optimal move and (2) understand the corresponding variants for winning the game. We use human experts for assessing the difficulty of Tsumegos, though checking with various thinking times of a program could provide estimates as well (at least if we can know the rank of the program for a given thinking time). All Tsumegos in this section were generated in a few days on an Intel Celeron 2.2 GHz.

3.1 Killall-Go

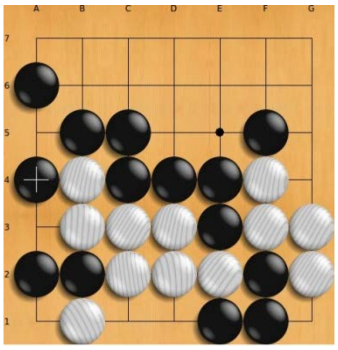

Fig. 1: Gnugo Level 6 vs MoGo 100 000: estimated Taiwan 1-2 Dan. White should play A3, Gnugo Level 6 played D1 and lost. (komi=48.5, i.e. Killall-Go).

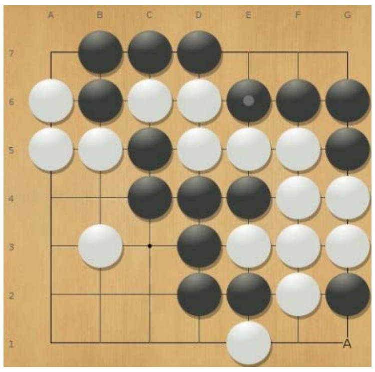

Fig. 2: Gnugo Level 10 vs MoGo 100 000: estimated Taiwan 1-2Dan. White should play B4 (komi=48.5, i.e. Killall-Go).

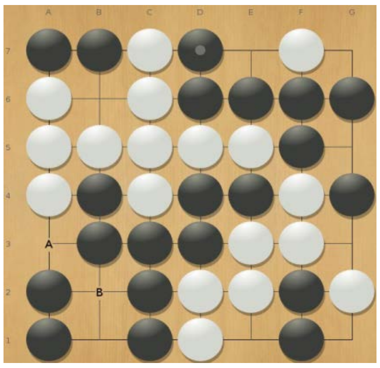

Fig. 3: Human amateur (~10 kyu) vs MoGo 100 000: estimated Taiwan 4-6kyu. White should play B2 and win by Seki. This Tsumego helped the corresponding human amateur to concept of Seki.

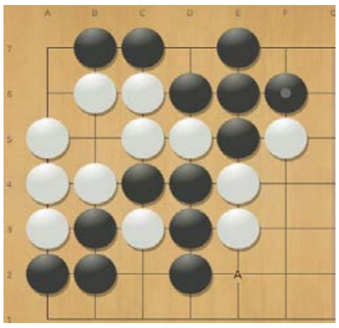

Fig. 4: Two Tsumegos, generated by games between Gnugo Monte-Carlo level 1 (as weak-player) and MoGo 100 000, Killall-Go. On the left: Estimated Taiwan 3-4 Dan; White should play F5 – however, a beginner might find this move by chance just because F5 matches a classical pattern.

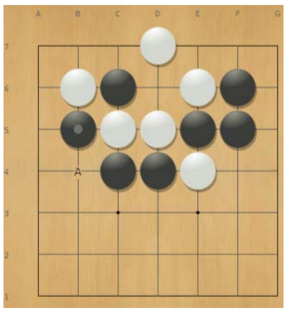

Fig. 5: Estimated Taiwan 4-6kyu. White should play B4. Here also, a beginner might find the right move by chance, because it matches a classical pattern.

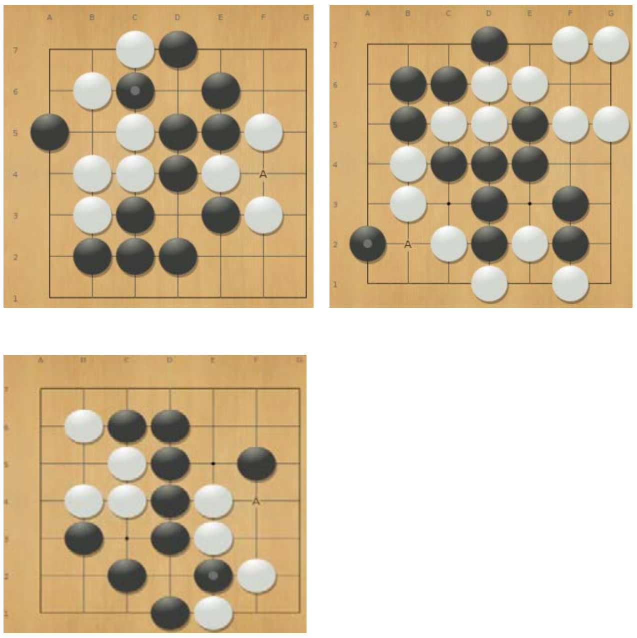

Fig. 6: Three Tsumegos, generated using Gnugo Monte-Carlo level 1 as a weak player

3.2 7x7 Go

It is widely assumed that the fair Komi in 7x7 Go is 9, which means that with komi 8.5,

Black should win. Therefore, we can apply STG for generating Tsumegos as follows:

- Game = 7x7 Go with Komi 9.5.
- Weak player = MoGo with 100 000 000 simulations per move and automatically generated opening book (generated by Meta-MCTS [1]).
- Strong player = MoGo with 100 000 000 simulations per move and handcrafted opening book (designed from the games in Taipei 2011 and Brisbane 2012 [2], using discussions with professional players and strong amateurs).

Incidentally, this was helpful for checking the automatically generated opening book.

This generated (in roughly 48 hours, Intel Celeron 2.2 GHz) the following Tsumego: Black should play E2 to win. Importantly, this move is difficult, even for a professional player; the optimal move is E2 in Chinese rules.

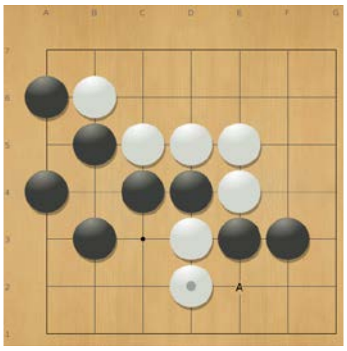

Fig. 7: A difficult Tsumego (Black to win, komi 8.5)

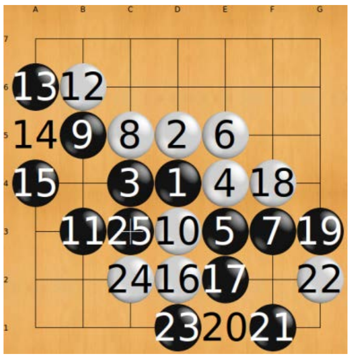

Fig. 8: Solution by Ping-Chiang Chou (5P).

4. Conclusion and Future work

4.1 Conclusion

We proposed a simple algorithm for generating problems. This algorithm can be used in the following case:

- There should be a safe strategy.
- There is a solver strong enough for checking where is the mistake in a game.
- We can design a player roughly of the expected strength of the Tsumego.
- Its outputs are problems: (1) at the level of the weak player; (2) not necessarily at endgame level;(3) accompanied with a solution; (4) as natural as the game style of the weak and strong player.

Some by-products of this work:

- including the solution to the 7x7 Tsumego with komi 9.5, we get a 7x7 Go player which never lost games, among many trials against strong MCTS programs:
- before including Ping-Chiang's correction, there was an opening for White leading to many losses for Black, in spite of komi 8.5, even for 10^8 simulations per move;
- after including Ping-Chiang's correction, there were still losses, for permutations of this opening;
- after including these permutations, we got no loss with 10^7 and 10^8 simulations per move (out of 64 and 73 games played as Black and White respectively, including 24 as Black against the specific opening on which MoGo was losing as black before Ping-Chiang correction);
- however we still get 18% and 9% loss respectively for Black and White, when playing with 10000 simulations per move, so the opening book is not enough for making the game trivial.
- several Tsumegos of various amateur levels (including seki situations);
- one Tsumego of strong amateur or professional level.

4.2 Future work

STG is here applied to Go, but it can be applied to other games, and also to other problems. The strong player decide the faults (with a constraint such as “no more than K faults”), the weak player makes the operator decisions, then the equivalent of a Tsumego is a situation in which the optimal decision by the human operator is not trivial. There is for sure a gap between our tests, in this paper, in a game, and an industrial application, but conceptually nothing prevents such a transfer, and as we have access to such simulators and solvers this further work is our priority.

Acknowledgements. The authors are grateful to NSC for support NSC 100-2811-E-024-001, to Bielefeld Search methodologies seminar, to BIRS game theory seminar, to Dagstuhl's seminar on Theory of Evolutionary Computation.

References

1) C.-W. Chou, P.-C. Chou, H. Doghmen, C.-S. Lee, T.-C. Su, F. Teytaud, and O. Teytaud, “Towards a solution of 7×7 go with meta-MCTS,” in Proceedings of Advances in Computer Games 2011.
2) Chang-Shing Lee, Olivier Teytaud, M.-H. Wang, S.-J. Yen. Science meets the game of Go, Computational Intelligence Magazine,

accepted.

3) T. Wolf, “Investigating Tsume-Go problems with RisiKo,” in Heuristic Programming in Artificial Intelligence 2, D. N. L. Levy and D. F. Beal, Eds. Chichester, England: Ellis Horwood Limited, 1991, pp. 153–160.
4) T. Wolf, “The program GoTools and its computer-generated Tsume-Go database,” in Proceedings of the First Game Programming Workshop in Japan, Hakone, Oct. 1994, pp. 84–96.
5) T. Wolf and Y. Seo, Unseen hard Life & Death Problems, 2011, unpublished.
6) B. Helmstetter, C.-S. Lee, F. Teytaud, O. Teytaud, W. Mei-Hui, and S.-J. Yen, “Random positions in Go,” in Computational Intelligence and Games, Seoul, Korea, Sep. 2011. Available: http://hal.inria.fr/inria-00625815
7) J. M. Robson, “The complexity of Go,” in Proceedings of the IFIP 9th World Computer Congress on Information processing, 1983, pp. 413–417.
8) M. Crâsmaru and J. Tromp, “Ladders are PSPACE-complete,” in Computers and Games, Lecture Notes in Computer Science, T. Marsland and I. Frank, Eds. Springer Berlin / Heidelberg, 2001, vol. 2063, pp. 241–249.