2012 Fifth International Symposium on Parallel Architectures, Algorithms and Programming

Parallel Monte Carlo search for imperfect information game Daihinmin

Junji Nishino
Department of Informatics
Graduate School of Informatics and Engineering
The University of Electro-Communications
Chofu Tokyo, Japan
Email: nishinojunji@uec.ac.jp

Tetsuro Nishino
Department of Informatics
Graduate School of Informatics and Engineering
The University of Electro-Communications
Chofu Tokyo, Japan
Email: nishino@uec.ac.jp

***Abstract*—Computer games, such as chess, have recently become an important research topic in the field of computation intelligence. Monto Carlo massive searching is very successful in the game of GO as it has a huge intractable search space and no good partial evaluation heuristics. Imperfect information games also have intractable properties, such as large scale search space, less heuristics, and unknown state information. Monte Carlo search is therefore effective to make computer players for the imperfect information game. There is always a time limit for using computational intelligence in actual applications. Such constraints require parallel processing to be speed up the Monte Carlo simulations.**

We introduce a parallel Monte Carlo searching method for imperfect information card game Daihinmin, a familiar card game in Japan. Computer Daihinmin competitions have been held since 2006. We present two kinds of parallelization algorithms for Monte Carlo computer Daihinmin players.

***Index Terms*—Monte Carlo search; Imperfect information game; Multiplayer game; Parallel processing; Tree search**

I. Introduction

In this paper, we introduce a parallel processing application for Monte Carlo searchs of imperfect information game trees.

A Monte Carlo searching has been widely adopted to solve large size problems such as game of GO player. It is recognized as a powerful tool because it needs no heuristics parameters and uses only simple repetitions. There are several parallel processing to solve complete information games [1], [2].

An imperfect information multi-player game, such as a card game, is a difficult problem to solve. A Monte Carlo search is therefore effective to make imperfect information game computer players [3]. There is always a time limit for computer game competitions. The limit stands for realistic constraints on using computational intelligence in actual applications. These constraints require parallel processes to be speed up the Monte Carlo simulations.

Daihinmin is a traditional, widely played Japanese card game. Its rules are very simple and a game goes by very quickly, thus almost all people play Daihinmin game in Japan. Computer Daihinmin Competition have been held since 2006, hosted by the University of Electro-Communications Japan. This article aims to show the adaptation of parallelized Monte Carlo algorithms for computer Daihinmin players.

The following sections presents the main problem in adopting Monte Carlo methods of selecting imperfect information game players. In section three, we describe two algorithms for a parallelized Monte Carlo search. In section four, an evaluation and comparison of the parallelized Monte Carlo algorithms is shown.

II. Monte Carlo Search for Imperfect information games

Imperfect information games, such as card games, have some ambiguity of game situation. We cannot therefore search and solve them precisely. An ordinary Monte Carlo search algorithm also cannot be applied to imperfect information games because the algorithm requires a game tree. To avoid this problem, a Monte Carlo search for imperfect information game is realized with Monte Carlo sampling.

First, we describe an extensive imperfect information games. Then we explain the Monte Carlo search algorithm and Monte Carlo sampling.

A. Extensive imperfect information game

Imperfect information game like as card game is a model of multi player correlations in our daily life. One of the most popular imperfect information games is card games. Card games have some imperfect information characteristics. For example, in card games, we cannot see opponents hands (cards).

If computer players could see their opponents cards, card games could be denoted in an extensive game tree and its Nash equilibrium optimal move can be calculated generally with searching methodology. Unfortunately, they cannot see their opponents cards. The computer players need to make an assumption about their opponents cards. This assumption process is called Monte Carlo sampling and is described in detail in a later section.

B. Monte Carlo search

A Monte Carlo search is a widely used searching technology. In particular, the UCT algorithm [4], a Monte Carlo tree search, shows very good efficiency on GO game computer players.

2168-3034/12 $26.00 © 2012 IEEE
DOI 10.1109/PAAP.2012.8

In decision making for game problems, searching is a powerful tool too. However, searching cannot process large problems, such as GO game. Iterative deepening depth-first searching is ordinarily used.

In a limited time, upper confidential bound (UCB) [5], [6] give us indexes that tell the next appropriate playout search node in a Monte Carlo search. UCB is defined by Equation (1) where n_j is the number of repetitions in the node j, $\overline{X}_j$ is the expectation as move j selected, and c is a control parameter.

$$\overline{X}_j + c\sqrt{\frac{2log\, n}{n_j}} \tag{1}$$

The first term is the expectation of point ratio of the $node_j$, and it tends to be selected in a random search playout. The second term is an evaluation value of how many selected the node. This evaluation function, therefore, causes to visit less selected nodes.

C. Monte Carlo sampling for imperfect information game

Monte Carlo sampling is needed to produce possible solutions for imperfect information games. In imperfect information games, the game cannot be rewritten into tree formulation because there is lack of information regarding their opponents' situations. This means that we cannot make a search tree for decision making. Candidate card distribution has to be made randomly for searching anyway.

Monte Carlo sampling is the assumption that opponents' cards are distributed randomly. If there exists some reliable constraints, not showing used cards, avoiding card duplication, etc., sampling is done to satisfy those constraints. Once sampling is finished, there are players cards hypothetical information. We can run a tree search on the sampled situation.

For example, in a five player card game with 53 cards, the players have 10 cards and situation candidates are 2.6×10^{23} patterns. In general, when each player has N cards, there are $(4N)!/(N!)^4$ information sets. They produce different search trees. Some calculations on the number of information sets are shown in Table I.

TABLE I
SIZE OF INFORMATION SETS CALCULATED BY MONTE CARLO SAMPLING

Number of player cards	Information set
1	24
2	2.5×10^3
3	3.6×10^5
4	6.3×10^7
5	1.1×10^{10}
6	2.3×10^{12}
7	4.7×10^{14}
8	9.9×10^{16}
9	2.1×10^{19}

There are a large number of search tree candidates caused by Monte Carlo sampling. Althogh the UCT algorithm is very powerful for solving imperfect information games with Monte Carlo sampling, UCT cannot be easily adopted. The reason is that UCT needs concrete game tree that cannot be determined by Monte Carlo Samplings.

III. PARALLELIZED MONTE CARLO SEARCH FOR IMPERFECT INFORMATION GAMES

Imperfect information games have problems not only on the size of information sets, but also with the large scale search space for each information set.

For example, in a five player card game as above, when each player has N cards, the size of the search space is approximately $(N+1)!^5$. Some calculations are shown in Table II.

TABLE II
SIZE OF GAME TREE FOR EACH SAMPLING

Number of player cards	Tree size
1	32
2	7.7×10^3
3	7.9×10^6
4	2.4×10^{10}
5	1.9×10^{14}
6	3.2×10^{18}
7	1.0×10^{23}
8	6.2×10^{27}
9	6.2×10^{32}

Because the size of the search space is quite large, the Monte Carlo search must be appropriate for the problem. At the same time, parallelized processing is also a fit for the Monte Carlo search so that it require many times of simulation process,

The Monte Carlo search algorithm for extensive imperfect information games consists of two stages. The first step is Monte Carlo sampling and the second is actual Monte Carlo searching on the assumption candidate tree.

This structure cause us several parallelizing strategy. In this paper, we compare two kinds of parallelization:

1) leaf parallelization and
2) sample parallelization

with Monte Carlo sampling.

Parallelization requires a uniform sequence of processing for blocks of threads to make efficient calculations. This constraint set some conditions on the Monte Carlo search process. With regard to this efficiency, two strategies have different characteristics.

A. Leaf parallelized algorithm

Leaf parallelization is that N playouts from the same leaf are processed in parallel. N is the parallelizing factor that is the same size as the block. This parallelizing wes formulated by Cazenave and Jouandeal [7].

This is one of the simplest ways to parallelize. The algorithm is as follows.

1) Make a distribution candidate usingsa Monte Carlo sampling
2) Select a leaf with UCB
3) N parallel playout simulations on a leaf
4) Update the UCB for each leaf nodes

B. Sample parallelize

Sample parallelizing is a way that Monte Carlo sampling produces a playout. This algorithm needs to parallelize Monte Carlo sampling itself.

1) Make a distribution candidate using Monte Carlo sampling.
2) Assign a playout according to UCB
3) Go to 1 until N parallel playouts are set up
4) Process parallel playout simulations
5) Update the UCB for each leaf node

C. Parallel Monte Carlo search with GPGPU processing

GPGPU is a low-power, very high speed, and inexpensive high performance parallel processing device. Currently, we can easily obtain a GPGPU processor that has 200 through near 3,000 or more cores.

Some limitations of GPGPU are as follows : processing is restricted in block architecture, the branching program has difficulty realizing programs, and it cause severe inefficiency. GPGPU applications for searching must be simple.

IV. Experiment

We have completed experiments to show the performance differences of the two methods of Monte Carlo sampling and Monte Carlo searching.

A. Evaluation

The comparison evaluation value is the winning rate against baseline computer players. The winning rate of multi-player games depends on opponents mixture. In these experiments, other players are all baseline players.

There are many kinds of imperfect information games, and they are quite different from each other. In this paper, we use a popular Japanese card game Daihinmin, as the test game, as described below. Computer Daihinmin competitions have been held since 2006. Snowl is the 2010's champion client program, which uses a complex sampling assertion model and a Monte Carlo search. We have experimental games with five computer players and 53 full set of game cards. The one of the players is Snowl and second one is proposed test program, the other three is default client with server. The computer Daihinmin competition has a time restriction rule.

If many more simulations are done by the simple Monte Carlo player client than are done by Snowl, a simple player can beat Snowl. The winning ratio against the number of Monte Carlo simulations is shown in Figure 1.

B. The card game Daihinmin

Daihinmin is a widely played card game in Japan.

Its rules are as follows:

1) Use 52 cards and one or two Jokers.
2) 3 to 6 players.
3) Deal each players the same amount of cards (9 to 18).
4) Open with a leading card from the dealing player.
5) The opening player chooses the type of play — single card, pairs, three cards, etc.

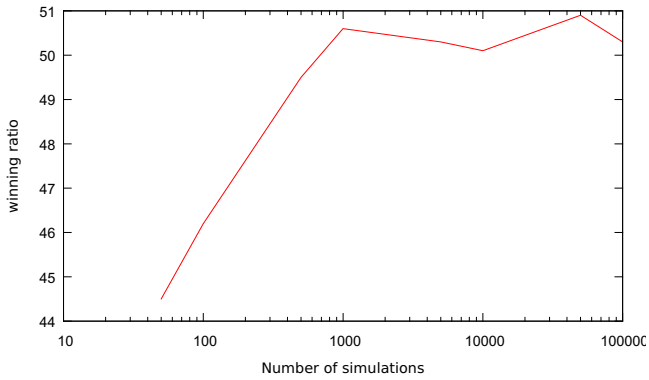

Fig. 1. Winning ratio to number of simulations for simple Monte Carlo vs. Snowl

6) The following player has to play the same type of cards stronger than opened cards or must hold (pass).
7) The weakest card is 3. The strongest card is 2. 2 is stronger than Ace. Between 4 through 10, greater is stronger. the Jack, Queen, King, and Ace are stronger according to this sequence.
8) If no one can play a stronger card than opened cards, then the set is finished. The last player becomes next opener.
9) Play continues until each player has played all his/her cards.
10) The winner is the first player to play all his/her cards.

The game is very easy learn and play; thus, it is the most popular card game in Japan. This game is a kind of imperfect information multi-player game from a view of game theory.

C. Leaf parallelization result

The winning rate of Leaf parallelized Monte Carlo versus Snowl is shown in Table III. Since every 128 random simulations on a block are done on the same leaf, the result in Table III starts from 500 simulations that is larger than 128.

TABLE III
WINNING RATIO OF THE NUMBER OF SIMULATIONS WITH LEAF PARALLELIZING

Number of simulations	Winning ratio
500	46.6%
1,000	48.4%
5,000	49.2%
1,0000	50.3%

D. Sampling parallelized result

Winning rate of the sampling parallelized Monte Carlo simulation is shown in Table IV.

E. Discussion on the results

Using sampling parallelization, the winning ratio reached over 50% in 1000 simulations in this experiment. On the other hand, using leaf parallelization, the winning ratio barely goes to 50% at 10,000 of simulations.

TABLE IV
WINNING RATIO OF THE NUMBER OF SIMULATIONS WITH SAMPLING PARALLELIZING

Number of simulations	Winning ratio
100	46.2%
500	49.5%
1,000	50.6%
5,000	50.3%
10,000	50.1%

Using the leaf parallelization algorithm makes producing an effective parallel program simple. However, its winning ratio is no better than using sampling parallelization.

In other words, this results indicate that leaf parallelization requires more simulation than does sampling parallelization to achieve the same winning ratio. If we increase the number of simulations, the performance is increased as total number of sampling and simulations.

The effective use of GPGPU requires simpler algorithms. Leaf parallelization is simpler than sampling parallelization. For GPGPU computation, leaf parallelization is suitable for Monte Carlo simulations.

V. Conclusion

Monte Carlo searching for games is a very powerful tool that can solve large scale perfect information games. It is also powerful enough to solve imperfect information games that are hard to handle in the ordinary way. The Monte Carlo search algorithm to solve imperfect information games is easy to parallelize and it may enable the creation of strong computer players.

In this paper, we compare two parallelization strategies experimentally. Sampling parallelization show better performance than leaf parallelization under parallel factor is 128. The parallel process in GPGPU consumes less energy and shows high performance. Considering the use of GPGPU, leaf parallelization is more suitable than sampling parallelization because of its simpler alborithm. We plan to speed up a Monte Carlo searching algorithm for imperfect information games using GPGPU.

References

[1] G. M.-B. Chaslot, M. H. Winands, and H. van den Herik, "Parallel monte-carlo tree search," in *CG 2008 LNCS 5131*, 2008, pp. 60–71.

[2] J. Long, N. R. Sturtevant, M. Buro, and T. Furtak, "Understanding the success of perfect information monte carlo sampling in game tree search," in *Proceedings of the 24th. AAAI Conf.* AAAI, 2010, pp. 134 – 140.

[3] G. Tesauro and G. Galperin, "On-line policy improvement using monte-carlo search," in *Advances in Neural Information Processing*, vol. 9, 1996, pp. 1068–1074.

[4] N. Sturtevant, "An analysis of uct in multi-player games," in *Computers and Games*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 5131, pp. 37–49.

[5] P. Auer, N. C. Bianci, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 2002.

[6] L. Kocsis and C. Szepesvari, "Bandit based monte-carlo planning," in *the 17th European Conf. on Machine Learning*, 2006, pp. 282 – 293.

[7] T. Cazenave and N. Jouandeau, "On the parallelization of uct," in *Proceedings of the Computer Games Workshop 2007*, e. van den Herik, Ed., 2007, pp. 93–101.