Speeding Up Gradient-Based Algorithms for Sequential Games*

(Extended Abstract)

Andrew Gilpin
Hg Analytics, LLC
New York, NY, USA
andrew@hganalytics.com

Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
sandholm@cs.cmu.edu

ABSTRACT

First-order (*i.e.*, gradient-based) methods for solving two-person zero-sum sequential games of imperfect information have recently become important tools in the construction of game theory-based agents. The computation time per iteration is typically dominated by matrix-vector product operations involving the payoff matrix A. In this paper, we describe two techniques for scaling up this operation. The first is a randomized sampling technique that approximates A with a sparser matrix $\tilde{A}$. Then an approximate equilibrium for the original game is found by finding an approximate equilibrium of the sampled game. The second technique involves the development of an algorithm and system for performing the matrix-vector product on a *cache-coherent Non-Uniform Memory Access (ccNUMA)* architecture. The two techniques can be applied together or separately, and they each lead to an algorithm that significantly outperforms the fastest prior gradient-based method.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Economics

Keywords

Equilibrium finding, automated abstraction, computational game theory, sequential games of imperfect information

1. INTRODUCTION

Game theory is increasingly serving as the foundation on which many successful game-playing agents are based. This is perhaps most noticeable in the development of agents for Texas Hold'em poker where virtually all of the best poker-playing programs are based on game theory [3, 5, 6]. The primary limitation in improving the performance of the agents is the scalability of equilibrium-finding algorithms.

*This material is based upon work supported by the National Science Foundation under ITR grants IIS-0427858 and IIS-0905390.

Cite as: Speeding Up Gradient-Based Algorithms for Sequential Games (Extended Abstract), Andrew Gilpin and Tuomas Sandholm, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

In this paper, we focus on the matrix-vector product operation that accounts for the majority of computation time in gradient-based algorithms for solving games. We propose two techniques for speeding up this operation. The first technique is based on approximating the payoff matrix of the game via a randomized sampling procedure. The second technique is an algorithm and system for performing the matrix-vector product on a *cache-coherent Non-Uniform Memory Access (ccNUMA)* architecture. Fully taking advantage of this hardware requires the algorithm designer to take special care in the design of the algorithm's memory management.

2. TWO-PERSON ZERO-SUM GAMES

A two-person zero-sum game is specified as a matrix A and strategy sets Q_1 and Q_2 for players 1 and 2, respectively. If player 1 plays strategy $x \in Q_1$ and player 2 plays strategy $y \in Q_2$, the utility to player 1 is $x^{\mathrm{T}}Ay$. (The utility to player 2 is the negative of player 1's utility.) The equilibrium problem is to find a solution to

$$\max_{x \in Q_1} \min_{y \in Q_2} x^{\mathrm{T}}Ay = \min_{y \in Q_2} \max_{x \in Q_1} x^{\mathrm{T}}Ay. \quad (1)$$

A solution (x^*, y^*) to the above equation is called a *minmax solution*. If the solution is within ε of optimal, it is called an ε-equilibrium.

When Q_1 and Q_2 have linear descriptions, as is the case for matrix games and sequential imperfect information games, Eq. (1) can be solved using linear programming. However, standard linear programming algorithms exhibit poor performance in practice when applied to that equation, especially with respect to the amount of memory required. To address the memory issues, the *excessive gap technique* [4] has been adapted for solving sequential games of imperfect information [1]. That algorithm finds an ε-equilibrium in $O(1/\varepsilon)$ iterations. Another algorithm that was specifically designed for solving games uses a simple iterative smoothing technique to find an ε-equilibrium in $O(\log 1/\varepsilon)$ iterations [2]. Both algorithms rely heavily on the matrix-vector product operation for computing gradients of the objective functions. The techniques we develop in this paper are applicable to any algorithm that requires performing a matrix-vector product using the game's payoff matrix and a strategy vector. This includes any gradient-based algorithm.

3. SAMPLING

For a given game, let Θ denote the possible sequences of chance moves. In the sequence form representation of zero-sum games, each leaf in the game tree corresponds to a non-zero entry in the payoff matrix A. Each leaf also corresponds to a particular sequence of chance moves. Thus, we can partition the non-zero en-

tries of A into a collection of non-overlapping matrices $\{A_\theta\}_{\theta\in\Theta}$ and we can calculate the matrix-vector product as a sum of products of these matrices:

$$Ay = \sum_{\theta\in\Theta} A_\theta y.$$

Instead of evaluating $A_\theta y$ for all $\theta \in \Theta$, we can estimate Ay by evaluating the products only for a small, randomly-sampled subset $\tilde{\Theta} \subset \Theta$:

$$Ay \approx \sum_{\theta\in\tilde{\Theta}} z_\theta A_\theta y.$$

The z_θ in the above equation are normalization constants that depend on the specific sampling performed, and are computed by simply making sure that the sampled probabilities sum to one.

We developed the following algorithm for performing dynamic sampling.

Step 1 Initialize $p = 0.01$ and initialize (x,y) arbitrarily.

Step 2 Randomly generate the sample $\tilde{\Theta} \subset \Theta$ so that $|\tilde{\Theta}| = \lceil p|\Theta| \rceil$.

Step 3 Run the gradient-based algorithm starting at (x,y) until overfitting is detected (if $p = 1$ then run the gradient-based algorithm indefinitely).

Step 4 Let $p \leftarrow \min\{2p, 1\}$.

Step 5 Go to Step 2.

In an experiment, we showed that for a large instance of an abstraction of Heads-Up Limit Texas Hold'em poker, it took the non-sampled version of the algorithm 32 hours to reach a gap of 20, whereas the sampled version only takes 3.7 hours to achieve the same gap.

4. EXPLOITING CCNUMA ARCHITECTURE

There is a rapidly accelerating trend in computer architecture towards systems with large numbers of processors and cores. At the highest end of the computing spectrum, this is illustrated by the development of the *cache-coherent Non-Uniform Memory Access (ccNUMA)* architecture. A NUMA architecture is one in which different processors access different physical memory locations at different speeds. Each processor has fast access to a certain amount of memory (near it in practice), but if it accesses memory from another physical location the memory access will be slower. However, all of the memory is addressable from every processor.

A NUMA architecture is *cache-coherent* if the hardware makes sure that writes to memory in a physical location are immediately visible to all other processors in the system. This greatly simplifies the complexity of software running on the ccNUMA platform. In fact, code written using standard parallelization libraries on other platforms will usually work without modification on a ccNUMA system. However, to fully take advantage of the performance capabilities of the hardware, our matrix-vector multiplication algorithm for games needs to be redesigned somewhat. Since the matrix-vector product accounts for such a significant portion of the time (this is true even when using sampling as discussed in the previous section), developing an algorithm that scales well in the number of available cores could have a significant impact in practice.

Processors in a ccNUMA system have fast access to local memory, and slower access to memory that is not directly connected. Thus, it would be ideal for every processor to primarily access data from its local memory bank. The ccNUMA system maps virtual memory to physical memory based on which processor first writes to a particular memory location. Thus, a common technique when developing software for a ccNUMA platform is to include an initialization step in which all of the threads are created, and they allocate their own memory and write initialization values to every location in their own memory.

We use this approach in our algorithm. Given that we have access to N cores, we partition the matrix into N equal-size pieces and communicate to each thread which segment of the matrix it should access. Then each thread allocates its own memory and loads the pertinent information describing it's submatrix. Thus, each thread will have the most pertinent data loaded in the physical memory closest to the core on which it is running.

We developed and tested our algorithm on a ccNUMA machine with 768 cores and 1.5 TB RAM. (In our experiments we only had access to 64 cores.) We tested the time needed for computing a matrix-vector product for an abstracted instance of Texas Hold'em poker. We compared our algorithm against the standard parallelization approach that does not take into account the unique physical characteristics of the ccNUMA architecture. Our new approach is always faster, and at 64 cores it is more than twice as fast.

5. CONCLUSIONS

In this paper we introduced two techniques for speeding up any gradient-based algorithm for solving sequential two-person zero-sum games of imperfect information. Both of the techniques decrease the amount of time spent performing the critical matrix-vector product operation. We developed techniques based on randomized sampling for quickly estimating a gradient. We also specialized our software for running on a ccNUMA architecture, which is becoming ubiquitous in high-performance computing. Experiments showed at least a 2x speed improvement over the standard parallelization approach for up to 64 cores. The two techniques that we developed can be used together or separately.

6. REFERENCES

[1] A. Gilpin, S. Hoda, J. Peña, and T. Sandholm. Gradient-based algorithms for finding Nash equilibria in extensive form games. In *3rd International Workshop on Internet and Network Economics (WINE)*, San Diego, CA, 2007.

[2] A. Gilpin, J. Peña, and T. Sandholm. First-order algorithm with $\mathcal{O}(\log(1/\varepsilon))$ convergence for ε-equilibrium in games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2008.

[3] A. Gilpin, T. Sandholm, and T. B. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas Hold'em poker. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[4] Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal of Optimization*, 16(1):235–249, 2005.

[5] M. Zinkevich, M. Bowling, and N. Burch. A new algorithm for generating equilibria in massive zero-sum games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007.

[6] M. Zinkevich, M. Bowling, M. Johanson, and C. Piccione. Regret minimization in games with incomplete information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.