

KnightCap Parallel Chess Program

A parallel chess program called KnightCap has been developed for the AP1000+. The program was developed in order to explore several interesting research issues within the framework of a real application. The program has several notable features, including a novel efficient parallelisation methodology, a learning algorithm and a 3-dimensional rendered interface.

Research Group

Andrew Tridgell
Jon Baxter

Objectives

The KnightCap project brings together several research interests in a single application program, allowing ideas to be tested in a more interesting way than can be achieved with traditional benchmarks. In particular, the program provides:

- * An interesting parallelisation task. Chess is a highly irregular problem which is notoriously difficult to parallelise efficiently.
- * An interesting systems programming task. Chess programs put a lot of stress on the memory management systems of an operating system. This has led to the development of novel non-uniform memory management strategies which may be applicable to other tasks.
- * An interesting algorithmic research problem. The search and evaluation of chess game trees is a classic computer science problem, but there are still many unresolved issues and opportunities.
- * An interesting computer learning problem. Learning from experience is one of the most difficult tasks for computers. KnightCap gives the opportunity to explore learning algorithms with an easy method of checking results.

Status

KnightCap was started in February 1997. Since then it has grown to be quite a successful chess program, taking on players from around the globe on the Internet Chess Server, where it has won over two thirds of the 3200 games it has played.

Parallelisation

Computer chess algorithms are notoriously difficult to parallelise efficiently. The principle problem is that the tree that needs to be searched is not at all regular, but instead varies considerably along different branches. This is a result of the extension and pruning techniques that must be applied for competitive chess play.

The most obvious parallelisation technique is to use each CPU to search along different branches of the game tree. This turns out to work very badly.

In KnightCap a slightly different approach has been taken. The newly developed MTD(f) algorithm was implemented with each CPU searching the same position but using a different evaluation cutoff value. Combined with a global hash table this turned out to be a simple yet quite effective strategy. Speedups of approximately seven were achieved with a 16 CPU AP1000+.

Learning Algorithms

A competitive chess program normally needs a enormous amount of manual effort in tuning the evaluation function and opening book to achieve reasonable play. This makes it very attractive to try to automate this process using a learning algorithm.

We have implemented two types of learning algorithms in KnightCap. The first is a static position recogniser designed to replace the standard opening book. The algorithm updates the contents of its position database based on games played, using a alpha-beta search between games to keep the positions consistent.

The second type of learning algorithm is is variant of the TD lambda algorithm. This algorithm updates the evaluation coefficients that make up the positional evaluation knowledge in the program.

The learning algorithms are progressing well with encouraging results being shown after a relatively short learning time.

Media interest

Considerable media interest has been shown in KnightCap in Australia. So far there have been 3 television interviews, 2 radio interviews and a number of newspaper reports.

Plans and Prospects

It is expected that the algorithms will be developed more over the next few months. The main emphasis will be on the learning algorithms. It is hoped that this project will demonstrate the feasibility of evaluation learning for chess computers.