16.412J - Cognitive Robotics

Problem Set #1

Jeremie Pouly

Part A:

The three topics in reasoning applied to embedded systems I would like to learn more about are the following. The first two are related to the project I would like to pursue, and the third one to something I already partially study and that is a real key issue for incoming space exploration.

- **Search algorithms**: I don't know much about this topic which is the most important for chess players. Today's computers allow us to do amazing calculations and to explore tons of possibilities in very short times, and it is easy to forget that for even pretty simple real-life problems, the pool of answers in so huge that we can't just explore everything. The real issue when we want to solve a problem is not the capability of the computer to theoretically solve it, but rather the time it needs to solve it. Indeed if it will need six months to find an answer, it can be useless since the problem itself could change during these six months. This is even truer for potentially infinite problems such as what a chess player should play in a certain chessboard configuration. Most of the time there won't be any solutions that will lead to victory whatever the human player choose to play (100% probability victory). In such cases we have to decide when to stop the algorithm searching for the best move, knowing that it could be a potentially better move that has not been found yet. As the limit is usually given by the calculation time – actually it is the deepness of the search but this is chosen functioned of the calculation time – the real challenge is to find faster search algorithms. Thus the level of a chess player is firstly given by the search algorithm that it uses.

- **Machine learning**: by that I mean the ability of a computer program to evaluate from its original state to improve its performance. This is also a real issue in robotics and any AI system. Theoretically I believe that if we can develop a good enough learning process, we will be able to create human-like robot able to grow up from "childhood-like state" to adult state and that will really be unique and autonomous as we are. As far as current interest, I would like to learn more about this topic because this is the real essence of AI and robots. If we are to send robots on Mars, it would be a real improvement if those could act more by themselves and not wait all the time for the eight minutes communication delay with Earth. However as we don't really know the Martian environment, this could only succeed if the robots are able to learn alone from what they see and touch. Closer to my project, it can be useful to develop a chess player that can improved itself by playing against the same group of human and even against humans in general (for I think humans in general – with the exception of world champions – tend to play according to a certain scheme). It could allow the AI chess player to define its own heuristic process that will lead it to victory.

- **Communication with humans**: Future space exploration should involve human / robot cooperation since one day we will be bored to send only robots and we will like to bring the space exploration to another level, and because, at the same time, it's cheaper and safer to send robots than humans. Then if we want to use both robots and humans in cooperation, we must find a way for them to communicate while carrying a mission on Mars for example. The robots are currently seen as "extra-hands" for the astronauts, therefore they must be able to understand the orders and request the humans will tell them. This is the first part of communication and I already worked on this problem of voice recognition. I coded a Matlab program able to recognize a small dictionary based on voice characteristics. But at the same time robots should be able to address concerns, warning or simple answers to the astronauts: we would largely improve the cooperation if communication could go both ways. I would be glad to conceive a robot able to discuss with humans, play trivia or tell jokes and stories in a pretty fluent language – even limited to certain fields.

Part D:

I chose three papers on computer chess player dealing with both learning and search algorithms as it is the foundation of the AI that I will use. My last paper is not directly related to what I will do during my project, but is rather an opening to show what we can do with AI chess player beside only play normal chess.

"An Evolutionary Approach for the Tuning of a Chess Evaluation Function using Population Dynamics". Graham Kendall, Glenn Whitwell.

I chose this paper because it explains how to use computers real power, which is to be able to do the same thing hundreds of time in a really short time with no error, to find the best chess evaluation function. It has the advantage of giving an optimal evaluation function with a rather low cost. I also chose this article because it shows that even light modifications to the base of the chess player, like the weighing of the pieces, can really change the way the AI plays.

The paper presents how to use machine learning in order to find the optimal evaluation function just by comparison between all the possibilities. Instead of imposing a certain weighing to the evaluation function, the idea of the article is to use machine learning techniques to allow the computer to choose the best estimate of the optimal evaluation function within a random population. The algorithm first step is to generate the random population of evaluation function candidates. Then the members of the population compete against each other, to find out which one is the more efficient. When a candidate looses a game, it is discarded from the pool and a clone of the winning candidate is added to the population instead, in order to accelerate the convergence, until there is only one member left in the population. The authors point out that if we apply just a naïve method like that, we might end up with the best estimate of the optimal evaluation function within the population, but this may still be far from the optimal itself. In order to reach this objective, they present another way to replace the discarded candidates by a crossover evaluation function child of the two parents with weightings which depends upon the result of the game and the standard deviation of the parents' parameters. This allows us to find a final function, toward which the population is converging, closer from the optimal evaluation function. If the initial repartition of the population was fair enough, we will hopefully find the optimal itself. The final evaluation function that comes up at the end of this algorithm is then tested successfully against a commercial chess program which proves the efficiency of the method.
What I liked in this article is the idea that a simple algorithm can lead to a great improvement of the chess player. Indeed both the original idea to optimize the evaluation function and the way to do it are at the same time simple and smart and the results are great. On the other hand, there are maybe some points that need more explanations like the fact that the members of the population compete only against some other members and not all of them. Maybe one kind of weighting might be efficient against another one but worse in average… At the same time this is only validate for three moves ahead, which is pretty low for computer chess and one can think that it could for example help the bishops against the knights since the latter needs more moves to be efficient.
Compare to the two other papers, this article is like a first step as it chooses the coefficients that will be used in the search algorithm. It is therefore needed if we want the search to be efficient and it must be done before trying to improve the search itself.
It will apply this method to my chess player because it leads straight forward to the optimal evaluation function for the search algorithm.

"*New Advance in Alpha-Beta Searching*". Jonathan Scaeffer, Aske Plaat.

I chose this article because it describes the core of any computer chess player: the alpha-beta searching. Indeed most chess players use this pretty simple and brute algorithm. I found the paper interesting because it states how this algorithm works, what are its strengths and it weaknesses, and above all what can be done to improve it despite the years of researches already spent on it.

After recalling the importance of the Alpha-Beta algorithm in game tree search, the paper summarizes its evolution from the earlier version to the actual one. It points out four improvements that have already been done and then presents three new enhancements. The first four modifications that has been done is the use of transposition tables to reduce the complexity of the tree by recognizing previously visited nodes (cutoffs), and buy calculation time with memory resources. Then move ordering can maximize cutoffs effectiveness (using a transposition table) if we sort the moves at each nodes in a best-to-worst order instead of a depth order. The heuristic choice of the search window at the beginning of the search also helps a lot to eliminate many nodes that with a certain level of confidence shouldn't lead to the best solution. And finally, we can save resources using a variable search depth that allocates more resources to the promising nodes and leaves aside the weaken ones. The paper then proposes three new enhancements to the current version of the Alpha-Beta search used for computer chess. The first idea is useful if the search algorithm ends in a fail low - the value of the first move is less than the search window - because of an unlucky guess of the search window. The authors propose to restart the algorithm using the transposition table to seed a new search which seems to improve the quality of the search even if it is hard to evaluate the real value of the improvement. The second idea helps to narrow the search window with a guessing of the values that we should obtain at the end of the algorithm so that the calculations time is a lot smaller. It uses the evaluation function score from previous iterations, according to the MTD algorithm, which improves the algorithm by 9%. Finally the last enhancement is an attempt to maximize the benefits of the transposition table by doing additional lookups after cutoffs. This is called the Enhanced Transposition Cutoffs and it leads to a reduction of the search tree by 28%. The results are illustrated with experiments comparing current computer chess player with specific one that present these new enhancements and it appears that with the three new ideas presented in this article we may be able to reduce the search effort in chess by 35%.

The strengths of this paper are all the tricks it gives to improve the Alpha-Beta search – both the old and the new ones – but at the same time this looks a little bit like a laundry list. Indeed, there is no link between the tricks and the theoretical justifications are often dark. They seem to present empirical process that the reader has to believe rather than theoretical analysis.

This paper is dealing with the main aspect of the chess player which is the search algorithm and it is the continuation of the first paper in a sense that it finds some tricks to improve the search algorithm using the evaluation function designed in the first paper.

In my project, the results presented in this paper will be very useful because it will allow me to improve the efficiency of the Alpha-Beta search in order to construct deeper trees with less time resources.

"*Information-Theoretic Advisors in Invisible Chess*". Bud, Albrecht, Nicholson, Zukerman.

This article is a little bit different from the two first one because it won’t be useful directly for my project. However it gives a good idea on how to use chess as a test for decision making strategy in complicated real-life problems. I am really interested in chess player by themselves because it is fun to apply my computer science knowledge on a game, but at the same time I am also interested in creating a pseudo-AI able to play chess because it is the first step to develop more powerful AI for real-life robots. This paper is therefore really appropriated since it shows how we could use chess as a test for real-world strategic situations.

If games have some properties of real-world situations the main difference between chess and most of the real life problems is the level of uncertainty. Indeed, whereas real-life is full of uncertainty, chess are known to be one of the only games with no uncertainty at all. Therefore, in order to use chess as a test-bed for real-world problems, the article introduces the concept of invisible chess. This is basically a normal chess game except that each player has a certain number of pieces that his opponents can’t see. This is interesting because by choosing the number of pieces that are unknown, we can change the degree of uncertainty of the game – to match real-world situation – and as it is still chess we can keep a high strategic level. Since each player can’t see some pieces of the other player, if we were to use only normal chess trees to find the best move, there would be a combinatorial explosion. The authors propose to counterbalance the new strategic complexity of invisible chess by adding an Information Theoretic Advisor to the Strategic Advisor. Indeed the early results – tests using only the strategic advisor – tend to show that the player with more information about the game will win more often, which seems pretty obvious. We can therefore increase the chances of success by using a weighed combination of the two kinds of advisor. According to the results presented in the articles, the efficiency of the algorithm is better if we use a combination of the two advisors; however, if the weightings are too much in favor of one advisor the results become quickly worse.

Thus this paper proves that we can use chess to demonstrate results about more complicated problems closer to real-world situations. I liked this idea of using chess as a demonstration field for something (uncertainty in this article) that has nothing to do with this game, because it illustrates why chess has been such an important step in AI development. However, the paper misses some details about the strategic advisor they used and what could be the applications in real-world situations of these results. Moreover, I think that more experimental data on the advisors, with maybe more that only three ratios for the relative influence of the advisors could have brought interesting subtle differences in the results.

This paper is like an opening after the two first articles, showing that we can test with chess more that only strategic issues. This is the main reason why I chose it.

This paper won’t really help me in my project, but it could help me after my project if I want to use what I did in another context.

Part E:

During the last third of the class I would like to conceive a chess player applying improved search algorithms – based on my computer knowledge – and heuristic processes – based on my chess experience. I would also like to include machine learning to have the AI improving its level with games. This project could be done by a group of students designing the core of the AI together, but each developing his own version of the chess player so that we can have them play against each other at the end of the term to see the relative efficiency of the heuristic processes.