Chapter 1

Real-Time Strategy High-level Planning

1.1 Abstract

This paper will give a fast overview of the current real-time strategy (RTS) plan making algorithms used in commercial games and in academic research. It will explain why RTS is an important and hard area to design Artificial Intelligence for. It discusses in short the strong and weak points of dynamic scripting, Monte Carlo planning and Case Based Planning. These techniques should eventually lead to an AI system that can outperform experts in this domain. Although that is still far away.

1.2 Introduction

Artificial Intelligence is being used since the very first computer games. In the past decade the popularity of computer games has increased and the revenue of the gaming industry exceeds multiple billions of dollars [Buro, 2003]. A certain genre of computer games, real-time strategy (RTS), also increased in popularity. In this type of game, playing against other humans is much more popular than playing against the artificial opponents. This is because commercial producers of RTS have not been able to enhance the Artificial Intelligence capabilities of their games without spending a large amount of money. Furthermore, the CPU time available for real-time strategy AI is limited in commercial games. [M. Buro, 2003]
Designing AI for a real-time strategy game is complex. The domain has several issues that need to be addressed by the AI. These can be found in the next section. Apart from the issues mentioned there, the RTS domain is also complex because of the large amount of objects in the game. Objects are units, buildings or any kind of non-static terrain features, such as trees (they can be cut down). [Danny C. Cheng, 2004]
This paper will discuss the commercial techniques and the techniques currently being researched. To give an overview of the plan-making methods used in a real-time strategy AI that is designed to win in a fair way.

- The next section will discuss the domain of real-time strategy artificial intelligence, how does a RTS game work and what problems are there for AI's to address.
- The Commercial techniques section is about the only commercial technique used in current real-time-strategy games.
- The Academic techniques section discusses the techniques to win RTS games that are researched in academic world.
- In the Discussion section we discuss the importance of research in this area.

1.3 Real Time Strategy Games

Real-Time strategy games are computer games in which the player controls an army in real-time to destroy other players' armies. A simplified military simulation. [Buro, 2003] Typically an army first has to be built for which one needs resources and buildings. These buildings will have to be constructed. To construct stronger units the player needs to do some sort of technology research often available in certain buildings. This research takes time and resources to complete. Furthermore a player normally has only partial information about the world around his base and needs to discover the world as his units traverse it. All units and buildings have a certain line of sight in which the player can see enemy units or buildings.

The key to winning a game is to balance the army production with the gathering of resources and technology. Concurrently the player should send scouts to discover enemy bases, armies and resources to destroy them. When all enemy units and buildings have been destroyed the player has won.
The tasks in real-time strategy games can be split into three levels of abstraction.

Unit control

Control at the lowest level, here the player controls the specific unit. For example, he can order a unit to destroy an enemy unit. Or he orders one of his workers to build a tower. Units have a standard behaviour at this level, even for human players. This standard behavior is normally a simple script-based system. With rules like: ”if enemy in range, attack“ . Players can increase their strength by overriding the standard behaviour. For example by focusing attacks on individual enemy units to quickly reduce their numbers. [Michael Chung, 2004]

Tactical planning

Tactical planning consists of making plans on how to attack the enemy. For example, if a map features uneven terrain, placing ranged units on hills can increase their combat strength against melee combat units. This is because they would first have to climb the hill (this is in this example presumed to be slower than walking on even ground, but is of course domain specific). This level also includes tactics using formations or splitting up the army to attack different positions. [Michael Chung, 2004]

Strategic planning

Strategic planning includes all kinds of high-level decisions such as when to build an army, what units to build, when to upgrade the technology, where to expand bases and where and when to capture important strategic places such as resources.

This paper primarily focuses on techniques for strategic and tactical planning, because the strength of an AI is mostly determined by these levels. A decent unit control can be achieved with simple reactive agents consisting of a set of rules and corresponding behaviours. [Michael Chung, 2004] High level strategy RTS AI problems are hard to solve. Unlike chess, a game that can be solved using a variation of A* searching, real time strategy poses different more complex problems for Intelligent Agents. [Buro, 2003]

Problems in real-time strategy games that have been identified include: [Buro, 2003]

Resource management

Players need resources in order to build and upgrade units and buildings. A good resource management agent is important for any kind of real-time strategy AI. The gathering and balancing of resources can be done relatively well with a simple reflex agent, since the AI has all the relevant information. It can simply assign more units to a certain resource when it runs low. [Danny C. Cheng, 2004]

Decision making under uncertainty

In an envrironment where only partial information is available and a certain strategy does not always produce the same result,it is much harder to reason. To be able to recognize important events the game probably needs some sort of pattern recognition. [Timo Kaukoranta, 2003] Even with no concrete information available, the AI should be able to form realistic hypothesizes and plans to execute. [Michael Buro, 2004]

Spatial and temporal reasoning

The world in an RTS changes fast. The applicability of a certain strategy needs to be constantly reevaluated. Terrain analysis is of the utmost importance in RTS games. Having a good spatial representation of the game world can significantly improve the AIs tactics and strategies. [Kenneth D. Forbus, 2002] The hardest contraint for RTS AI is time. The world changes so fast that when an AI is calculating its strategy, it already became invalid.

Collaboration

Clearly lacking in RTS AI is collaboration. Computer players never work together with other players to overcome a strong opponent, while humans regularly form alliances to defeat stronger players. A good method for communication/observation between different AIs and humans is necessary.[Michael Buro, 2004] This sounds easier than it is, good collaboration means the ability to recognize the allied strategy and helping them in the right way.

Opponent modeling, learning

Even harder than finding the allied strategy is the ability to discover the strategy used by the opponent. Human players can spot weaknesses in a strategy and exploit them in a few games, while current AI players have no way of accomplishing this yet. Learning from previous experience and formulating applicable counter strategies is a problem to be addressed. [Buro, 2003]

Others

Other challenging AI problems are present in Real-Time Strategy games, such as path finding for large groups of units, but these techniques are not very important for a strategy module, so they are not further discussed.

Real time strategy games are a challenging area of AI research with a lot of problems to address. Finding solutions for these problems might offer interesting algorithms that may be applicable to the real world.

1.4 Commercial Techniques

Some commercial RTS games are very popular. Games like Age of Empires by Ensemble Studios or Starcraft by Blizzard have sold millions of copies all around the world. Even though all of these games have some sort of AI, multiplayer where two or more human players fight against each other, is far more popular. This can mainly be contributed to the lack of decent artificial intelligence in these games. [Buro, 2003]
Reasons why current commercial RTS AI fails to challenge human players are:

Predictability and the lack of spatial reasoning

Human players excel at finding weak spots in strategies and using the terrain to its limits and finding suitable strategies to exploit them. Commercial strategy AIs only spatial reasoning is in the path finding algorithms. [Kenneth D. Forbus, 2002] This weakness combined with their predictability makes it very easy to win against these kinds of opponents as a human.
The predictability of commercial AIs comes from the fact that all AI is scripted. Even though these scripts can be really complex with lots of conditions and random events the AI is not able win the game in a fair way even against mediocre human players.

Scripting

A scripted AI is essentially a very complex set of rules that are predefined by an expert in the domain. In RTS this means that all paths of attack, all buildorders and all tactics are static and predefined. A buildorder is a sequence of actions that are considered good. For example when a game starts, first build 5 resource gathering units and let them gather food (a resource needed for new units). A fully implemented script often has a way to tell the difference between the begin, mid and endgame, where different parts of the script are active. At the start of a game buildorders are needed, while in mid-game for example the AI has to focus on making an army and expanding, while in the late game the focus lies on conquering and destroying the enemy. We could call such a complete script a " Strategy“. Commercial programmers include a set of these strategies in their games to give the human player some diversity. Even though these scripts have several disadvantages as discussed below, they are still the only technique used in commercial RTS games. This is because it still does give a static and good performance and is quite hard to beat even with recent Academic techniques.

Hard to implement

The implementation of a scripted RTS AI needs several experts in the game that think of viable strategies. Even after implementation it has to be tested and tweaked thoroughly. In combination with limited funds for development in commercial games and the fact that the rest of the game has to be near completion before the AI programmers can start implementing, causes that the available time to make and test commercial RTS AI is limited. [Michael Buro, 2004]

Domain specific

After implementing a scripted AI for a certain game, the implementation is not applicable for any other RTS games. This forces game developers to go through the implementation cycle again when designing the next game. If games are very similar it might be possible to reuse some of the old code, but generally a new game equals a new scripted AI [Istvn Szita, 2007].

Game developers use other means to entertain the player. Commercial games usually ship with a few different scripted strategies, so that it seems the AI does not always do the same thing. Another method is cheating, they give the AI more information than it should have. For example information about the players army even though it is not visible, or complete map information even though they did not discover it yet. Even with all this extra information most AIs are still not strong enough to compete against expert human players. Game developers can give the AI additional advantages in terms of higher resource acquisition rate, or stronger units. [Various, 2002] Having to rely on cheating to even pose a threat to expert human players is a clear indication how weak the RTS AI in modern day commercial games really is. The AI created with these cheats can be quite formidable for casual players, often the best AI is made weaker for novice human players.

1.5 Academic Research

In the past few years some researchers have tried to improve the performance of RTS games. Unfortunately the number of researchers is not yet quite high, as RTS has not really been brought to light as the new AI challenge as of yet. A very popular and related area is robotic football. It is related since it also features real time decision making and planning. Robotic football however is also focuses a lot on other Artificial Intelligence challenges, such as visual recognition and robotic-movement functions. On purely tactical and strategic level robotic football is a lot less complex than even a simple RTS, as robotic football does not feature a economy, nor does it have interesting terrain features. Agents in robotic football also have almost complete information about the environment. The most important difference is in the amount of objects. RTS games feature such a large number of objects and states that new planning methods have to be explored. [Buro, 2003]

RTS games that have been used in research are Wargus and ORTS. Wargus is a Warcraft II clone based on a open source engine called Stratagus. It is a good candidate for AI research because its code can be easily modified and extended. [David W. Aha, 2005] ORTS is an RTS game engine specifically designed for research. It has some interesting features, for instance the server executes the actions given by clients and sends only the available information for that client back. In that way there is no way to cheat, as the client has no way to know more information than is given by the server. Furthermore there is no default unit behaviour. So the player will have to give all the low level unit commands.[Michael Buro, 2004]

1.5.1 Dynamic Scripting

Commercial RTS games use scripted AIs. One of the biggest disadvantages of a scripted AI is that it is always using the same strategy and is therefore predictable. To address this problem Istvn Szita [2007] focused on Dynamic scripting to bring more diversity into the game. Dynamic scripting is essentially a reinforcement learning technique for scripts. [Istvn Szita, 2007] Dynamic scripting automatically generates its strategy on the fly by selecting a viable tactic in the tactic database. Each of these tactics is predefined and domain specific. Every selectable tactic is given a certain weight. When a tactic is used against a player and it had a positive effect, its weight increases and vice versa. The sum of all the weights in the tactic database is constant, so a certain function is responsible for increasing and decreasing weights on tactics. [Marc J.V. Ponsen, 2005]

Against static (using the same tactics every match) opponents dynamic scripting develops a strong counter-tactics that can beat most static opponents. Dynamic scripting can address problems on all three abstraction levels, as opposed to some previous research that for example applied reinforcement learning to low level unit behaviour. [Marc J.V. Ponsen, 2005]

Advantages of dynamic scripting

Whereas static scripts may contain weaknesses that can be found and exploited by human players, a dynamic script automatically reduces the effectiveness of these exploits, possibly making the game more interesting for human players. A dynamic script as proposed by Marc J.V. Ponsen [2005] has a small disadvantage. If it is trained against static opponents its strategy converges to a static and predictable strategy. Losing its strong point of being interesting. To further increase the diversity in tactics Marc J.V. Ponsen [2005] proposes a certain diversity function that rewards the AI for finding effective tactics that significantly differ from the previous tactics. This way the adaptively of the strategy can even be increased as the AI considers more options, making them stronger. Typically, this technique leads to a more interesting game for human players. [Istvn Szita, 2007]

Disadvantages of dynamic scripting

Most research done in dynamic scripting uses static opponents to learn from. But it is much harder for an adapting script to become effective against human players since the opponent also adapts and finds weaknesses in the new strategies. Thus its effectiveness against other adaptive players can be doubted. [Istvn Szita, 2007] Furthermore, making dynamic scripts still requires experts and domain knowledge to fill the tactics database. Making a decent AI would still require a lot of implementation time, probably even more than static scripting.

1.5.2 Monte Carlo Planning

A different approach to the strategic planning problem is Monte Carlo planning. Monte Carlo planning essentially generates a set of simulations, for all possible actions. It then chooses the plan that corresponds with the best simulation for the player and acts upon it. Note that in an RTS game the number of possible actions can be very large, so large that it might be impossible for modern computers to look even a few steps into the future. That is why an abstraction for actions and states has to be found. Typically, a larger number of considered plans results in a better strategy. [Michael Chung, 2004]

A plan in Monte Carlo planning consists of a sequence of repeating high level actions, such as attack. A large amount of these plans is generated, the outcome simulated and the best plan found. To find this plan there needs to be some sort of evaluation function that rewards effective plans. The quality of this evaluation function is important for efficient good results, but a less good evaluation function can still perform good by increasing the search efforts. [Michael Chung, 2004] In essence the Monte Carlo planning algorithm focuses on adversarial planning. Anticipating what the enemy will do and constructing a viable counter strategy for it.

Advantages of Monte Carlo planning

An important advantage of Monte Carlo planning is that it is less dependent on domain experts to make predefined tactics. Only the high-level actions and evaluation function are domain specific, and the majority of the evaluation function can be reused. [Michael Chung, 2004]

Disadvantages of Monte Carlo planning

Monte Carlo planning introduces a lot of calculating. In large complex games the amount of calculating could limit the depth of search the Monte Carlo algorithm can do. To keep the amount of considered plans and depth the same, even higher abstraction levels would be needed. [Michael Chung, 2004] Another problem with Monte Carlo planning is that it does not learn from previous mistakes. If it is playing against a learning adversary, and he finds a winning strategy, it will not be able to improve itself.

1.5.3 Case Based Planning

Case Based Planning (CBP) is a planning implementation based on case based reasoning. Case Based Reasoning (CBR) is similar to the dynamic scripting method as it also looks at the past and learns from mistakes. CBR is based on states. It looks at past experience and calculates the best sub plan in the current state. Current case based planning AIs focus on tactical and strategic levels of operation with high level abstract state spaces. The CBP algorithm is given an initial goal: "winning the game“. This goal is than expanded to find the sub goals that need to be completed first.

The Case Based planning algorithm given by David W. Aha [2005] simply selects the best sub plan given the current state and goal. If the execution of the plan was successful and eventually leads to victory, the choice of the sub plan is considered good, and the heuristic value of that certain sub plan in that state is increased. Case Based planning does not do any adversarial planning, nor does it try to model the opponent. CBP also uses a database of predefined tactics. Initially, when learning what tactics should be used in different states, the algorithm picks a random one and remembers what outcome it has. Obviously, good tactics will get high evaluation values and vice-versa. [David W. Aha, 2005]

Instead of pre defining strategies, Manish Mehta [2007] explains a way of learning from observing experts that play the game. An expert plays the game against a opponent (human, or AI) while the game makes a trace of all the actions done by the expert. When the game is finished the expert will indicate for each action what goal he tried to complete with it. The algorithm by Manish Mehta [2007] then computes viable tactics in certain states. The CBP algorithm can learn from its mistakes by doing certain bad tactics less often, but it has no way of fixing the tactics itself.

A further improvement on the Case Based planning algorithm is given by Manish Mehta [2009]. They tried to fix exactly that problem. When plans fail, this information is remembered and when the system is not playing, it can calculate what the differences are between successful plans and failures. When playing a game, the failure detection agent uses failure-patterns to discover causes of plan failures. When a explicit cause of plan failure is discovered, the corresponding tactic can be fixed using a so-called plan-modification routine. [Manish Mehta, 2009]

The main difference between Case Based Planning and Dynamic scripting is the underlying idea. In Case Based Planning the strategy and tactics consist of sub plans and sub goals and it looks if the goals set are reached. In Dynamic scripting there is no such thing as goals. It simply selects the tactic that yielded the best results in the past. While the result of both algorithms may be the same, the underlying principle is very different. [David W. Aha, 2005]

Advantages of Case Based Planning

An advantage of case based planning is the implementation time, by allowing the experts to play the game and show the AI what good decisions are, simplifying the process of designing an AI. [Manish Mehta, 2007] CBP is able to form effective strategies against dynamic opponents and can provide some diversity as well. [David W. Aha, 2005]

Disadvantages of Case Based Planning

A disadvantage of CBP based AI is that it needs to be trained against opponents before it can become strong. The more complex and diverse the opponents are the longer it takes CBP to learn how to win. [David W. Aha, 2005]

1.5.4 Others methods

There are many more approaches and extensions to the methods mentioned here. An example is transfer learning using CBR and reinforcement learning. It focuses on the application of previous experiences and tactics in new unknown states or maps to reduce the training time needed for CBP algorithms to work efficiently. [Manu Sharma, 2006] Totally different RTS AIs or parts of

RTS AIs could be designed using other algorithms. Like deductive, abductive or probabilistic (using Bayesian networks) plan recognition, but will not be further discussed in this paper since they are not yet applied to RTS games. [Danny C. Cheng, 2004]

1.6 Discussion

The availability of a open source RTS AI development platform, ORTS, will probably give the RTS researchers a way to test results against each other. It is kind of strange to find that many researchers still using the game Wargus for research. Furthermore there really should be an international RTS AI tournament, like there is for robot soccer, to get some more attention to this domain.

Applicability in other domains

The American Military wants High-performance simulators for training military personnel. The level of command and control AI, in particular that of the RTS domain is lacking. [Michael Buro, 2004] To realistically train military personal in these areas, high performing AIs are needed. Furthermore, the problems addressed in section 1.3 can be applied to several other real-time domains, such as planning in real world scenarios.

Current robotic research primarily focuses on recognition of the world rather than planning. Robotic football, a research domain that has some similarities to RTS, has almost complete information and it's hardest challenges lie in knowing where you, your team and the ball are and calculating and communicating the strategy among the team. The strategy itself is a set of predictable attack paterns that could easily be countered by a human. Real-Time Strategy AI, could be applied here to find more viable attack patterns.

In general, the ability to plan in real-time is very important in a lot of domains. Which makes real-time strategy games a ideal test bed for complex real-world situations. [Buro, 2003]

Would it be possible to combine several planning methods?

To the best of my knowledge such a combination technique has not yet been used in the Real-Time Strategy domain, but it is certainly interesting what sort of results it would give. The biggest problems with such a technique is the amount of computing power it would need. Any of the academic techniques discussed in this paper already require vast amounts of computing power (except dynamic scripting). Combining them with for example a majority rule would require a separate calculation for each strategy and comparing them to find the similarities. Which in turn would require a non-trivial strategy comparer that can identify the underlying tactic in each sequence of actions and finding the most preferred tactic. Although such a strategy comparer is already partially available in the Case-Based Planning technique.

Combining different techniques on different command levels (as explained in section 1.3) has been done in research before though. The lowest level of unit control has a quite good performance even with simple reflex-agent scripts, so the already available scripting has been used in some research to provide the most basic unit control. It is imaginable that for example Monte Carlo planning could be quite effective at the tactical control level, where the effect of ones actions can be calculated quite efficiently since we reduced the state-space to the military control. While for example Case-Based Planning would take care of the high-level strategic plans. Such a strategy might take advantage of the strong points of both techniques while minimizing the weaknesses.

Many more combinations of techniques are possible and it would be interesting to see the results of these combinations.If combined in the right way it hopefully creates an AI system that uses certain algorithms where they have the best results. There is a chance, however, that certain AI modules would have their own "strategy“ that differs from the general strategy outlined by the system, which could result in conflicting priorities and overall worse performance.

1.7 Conclusion

In this paper we have discussed the various real-time strategic planning AI techniques currently in use in commercial or research projects. The most prominent are the commercially much used scripting techniques, an academic method that dynamically chooses between scripts, a Monte Carlo

planner and a Case Based Planning system. Pointing out which of these techniques is currently the best is quite hard. As there have been made some really advanced scripts in commercial games and the performance of some Case Based or Monte Carlo planners have yet to be proved in more complex games.
The current academic technique that is the most advanced would be Case Based planning since it already can adapt, change and enhance available strategies and can be extended with a adversarial plan recognition system. [Manish Mehta, 2007] That said, there have not yet been any real performance comparisons between dynamic scripting and Case Based planning. [David W. Aha, 2005] Neither have there been any comparisons between Monte Carlo planning and either of the other two. Although Monte Carlo does not learn from its own mistakes, it might become a formidable opponent.
Further research in this area will be needed before any of them come close to a human expert.

1.8 References

Buro, M. *Real-Time Strategy Gaines: A New AI Research Challenge.* 2003.

Danny C. Cheng, R.T. *Case-Based Plan recognition for Real-Time Strategy Games.* 2004.

David W. Aha, Matthew Molineaux, M.P. *Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game.* 2005.

Istvn Szita, Marc Ponsen, P.S. *Interesting Adaptive Game AI.* 2007.

Kenneth D. Forbus, James V. Mahoney, K.D. *How qualitative spatial reasoning can improve strategy game AIs.* 2002.

M. Buro, T.F. *RTS Games as Test-Bed for Real-Time Research.* 2003.

Manish Mehta, Santiago Ontanon, A.R. *Case-Based Planning and Execution for Real-Time Strategy Games.* 2007.

Manish Mehta, Santiago Ontanon, A.R. *Using Meta-Reasoning to Improve the Performance of Case-Based Planning.* 2009.

Manu Sharma, Michael Holmes, J.S.A.I.C.I.A.R. *Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL.* 2006.

Marc J.V. Ponsen, Hctor Muoz-Avila, P.S.D.W.A. *Automatically Acquiring Domain Knowledge for adaptive Game AI Using Evolutionary Learning.* 2005.

Michael Buro, T.M.F. *RTS Games and Real-Time AI Research.* 2004.

Michael Chung, Michael Buro, J.S. *Monte Carlo Planning in RTS Games.* 2004.

Timo Kaukoranta, Jouni Smed, H.H. *Role of Pattern Recognition in Computer Games.* 2003.

Various. *AI Game Programming Wisdom.* 2002.