A HTN Planner For A Real-Time Strategy Game

Jasper Laagland
j.laagland@student.utwente.nl

ABSTRACT
This paper presents the design, implementation and evaluation of a Hierarchical Task Network in a real-time strategy game. The method that is applied is Hierarchical Task Networks. The design is based on other planners for real-time strategy games. Possible HTN planners and real-time strategy games are researched. An overview of other planners that have been applied to real-time strategy games is presented and the advantages and disadvantages of HTN planning to these planners are discussed. The design is implemented with SHOP2 and tested on Total Annihilation Spring, an open-source real-time strategy game. This research is focused on the strategic level of the game, which includes the building of buildings and units and resource management. The implementation is tested against the built-in AI of TA Spring. The HTN planner performs better on the tested subjects and is proven to be more efficient than the built-in AI.

Keywords
Hierarchical task networks, planning, real-time strategy.

1. INTRODUCTION
Real-time strategy (RTS) games are a subcategory of real-time games. Which means as much as that the game play is continuous rather than turn-by-turn. Very often they are seen as simulation-based games, examples are The Sims, SimCity. RTS is the counterpart of turn based strategy. In turn based strategy players each get a turn to execute their plans after which it is the next players turn. In RTS all players are able to act simultaneously which makes it dynamic and more complex than turn based strategy.

RTS is a very interesting field for AI research, there can be hundreds of interacting objects at one time. Also, there are different subjects on which one can focus while creating an AI, these include how to win local fights, how to scout effectively, hw to build, attack, and defend. The current state of AI for these types of games is still bad [BUR04], although human players are not able to control multiple units at a time they can still beat an AI opponent. Further there are a number of features that are relatively easy to implement which have not been implemented until now, such as tools for unit balancing [BUR04].

This paper presents a new AI strategy for RTS games. The goal is to create a more efficient AI than the built-in AI which plans according to a build list. This paper focuses on the strategic level of RTS games, this includes the creation of buildings and units and resource management. The method which is presented is Hierarchical Task Networks, this has already been applied to many other games including turn-based strategy which –even though simpler- is similar to RTS. To test its performance the HTN planner is evaluated against an already existing AI.

2. RESEARCH QUESTIONS AND METHODOLOGY
For this research a real-time strategy game and a HTN planner are selected. There are several real-time strategy games available that contain an open-source language. Next an HTN planner is selected. The planner should be able to plan in real-time and a minimum delay. The HTN is modeled with information of current AI of real-time strategy games and other HTN planners created for other (similar) games.

There are only a few planners available which can be used. The chosen game and planner are not written in the same programming language. This means that they cannot be combined in one program, a form of communication must be set up to make interaction between these programs possible.

Since the time span of this research is relatively short not a complete representation of the world can be modeled for the HTN. Therefore a choice has been made to represent a part of the world. It should be possible to evaluate this part separately from other parts of the game. The implementation of the planner is based on that of earlier implemented planners.

The HTN planner is evaluated against the AI that is currently available in the game.

Research Questions:

- Which HTN planner is suitable and available for planning problems in real-time?
- Which real-time strategy game is available and has a built in AI?
- Which other planners have been applied to real-time strategy games?
- What are the (dis)advantages of the HTN planner to the already implemented planners?
- On which other planners or AI design can the design of the HTN planner be based?
- What is a possible design for the HTN planner?
- How well does the HTN planner perform in contrast to the built in AI?

3. REAL-TIME STRATEGY
In RTS games each player controls an army of choice, a base must be established and buildings and units must be built. Buildings are produced by a builder or a construction yard. There is no limit on the amount of units and buildings that can be produced. Further resources must be allocated, in most RTS games there are two resources: money and energy. Sometimes money is replaced by raw materials. An RTS game can be played through a internet/LAN connection or against the built in AI. Because RTS is so complex it is a very interesting research field. The following fields are researched in RTS games [BUR03]:

- **Resource management:** Players start with an amount of resources. Then they must provide themselves with enough resources to manage producing buildings and resources.
- **Decision making under uncertainty:** At first little is known about the enemy and map, therefore choices have to be made on assumptions on the enemy status.

- **Spatial and temporal reasoning:** Understanding of temporal relations of actions and terrain analysis.
- **Collaboration:** When playing with more than two players, teams can be created which must cooperate.
- **Opponent modeling, Learning**: Learning the effects of certain actions and predicting the actions of opponents.
- **Adversarial real-time planning:** Providing abstractions of actions in order to make planning possible.

The method that is applied to create AI for commercial RTS games such as Command & Conquer and Warcraft is scripting [PON05]. An agent follows a pre-written script and executes the tasks sequentially.

3.1 Previous Research

There is a reasonable amount of research done in the field of RTS towards creating AI. The following topics have been dealt with:

- Monte Carlo planning [CHU05]; Monte Carlo planning a abstract representation of the world is made. A number of plans is generated and the planner samples these plans, the one with the highest stochastical probability is selected and executed. Thus, before Monte Carlo planning is able to choose the optimal plan it has to do learning first.
- Case-based plan selection [AHA05]; This method was used for creating an AI that can beat random enemies by using case based reasoning. A number of variables is used to learn the best action at a certain time.
- Evolutionary Learning. [PON05]; This method makes use of dynamic scripting. A number of scripts is created which the AI can select from. An weight update function distributes weights to these scripts and the one with the highest weight is selected.

3.2 Levels of Planning

For this research one RTS game is selected for which a HTN planner is created. There are several open-source RTS games available, e.g. OpenRTS, Warzone 2100 and Total Annihilation Spring. These games all share the characteristics as explained earlier in this chapter.

In RTS games there are different levels of planning [CHU05]:

Unit control: This is the lowest level, it consist of the behavior of the individual units. This includes tasks as path finding and attacking other units.

Tactical: This is the mid-level, this involves the control of groups of units, attacking strategies and placement of newly produced buildings.

Strategic: This is the highest level of planning. This level includes resource management, long term strategic plans. Also the building strategy is part of this level.

For this research it is not possible to explore all these levels. Therefore a choice has been made, the strategic level is selected for this paper. This can be evaluated separately by measuring the number of buildings and units and the resources over a time period.

4. HIERARCHICAL TASK NETWORKS

In HTN planning the world representation and the action representation are very similar to the STRIPS formal language. In both planners the states in the world are represented as a collection of atoms, the major difference between STRIPS and HTN is that the operators in STRIPS are atomic in contrast to the HTN in which operators can be complex. This allows the operators in HTN to be decomposed further into sub-operators. These can also be decomposed which results in a tree.

There are three types of tasks in HTNs:

- Goal states, properties that are wanted to be true.
- Primitive tasks, tasks that can directly be achieved by an corresponding action. These are actions like, move or build.
- Compound tasks, these involve multiple goal tasks and several desired changes in the world that cannot be expressed in single primitive tasks or goal tasks. For example eliminating an enemy base requires several actions (building units, attacking etc) before the task is achieved.

These tasks are connected through a network (hence the name hierarchical task network). A HTN planner resolves a plan as follows [ERO94]:

1. Input a planning problem **P**.
2. If **P** contains only primitive tasks, then resolve the conflicts in **P** and return the result of the executed tasks. If the conflicts cannot be resolved, return failure.
3. Choose a non-primitive task *t* in **P**.
4. Choose an expansion for *t.*
5. Replace *t* with the expansion
6. Use critics to find the interactions among the tasks in **P**, and suggest ways to handle them.
7. Apply one of the ways suggested in step 6.
8. Go to step 2.

Steps 3-5 is accomplished by choosing an appropriate reduction (replacing the non-primitive tasks with the corresponding primitive tasks in the network). When all non-primitive tasks are replaced a totally ordered plan must be found that satisfies all constraints. After each reduction a set of critics is checked to recognize and resolve interactions between the reductions. This is useful because this enables the detection of interactions early in the tree, so that backtracking is reduced.

HTNs have already been successfully applied to several other planning problems in computer games. Smith et al. developed the Brigde Baron [SMI98], a HTN planner which could play bridge against other AI programs. Munoz-Avila and Fisher have successfully applied HTNs to Unreal Tournament Bots [MUN04].

4.1 Advantages of Using HTN Planners

The main advantage of using HTN planning is that it will find a plan when there is a plan available. When a conflict is found in decomposing a task it automatically backtracks and tries to find the solution on a different decomposition. Unlike other planners which are not able to backtrack or stop at a certain depth.

The advantage of the tree structure of HTN planners is that a solution can quickly be found, HTN planners use depth first search. In contrast to other planners HTN planner does not need

to search the complete space to find a solution for the given problem.

An advantage to using case based reasoning is that with HTNs different strategies can be implemented. Case based reasoning uses past solutions of problems to solve new problems. Although this will require more work than just generate the possible actions and try to find an optimal solution. In case based reasoning approach an optimal strategy is chosen based on a number of selected variables which are the current state of the world [CHE04]. The same goes for the monte carlo approach, an optimal model is learned to selected the best action in a certain state in the world.

4.2 Disadvantages of Using HTN Planners

A downside of using HTNs is that when the world gets very complex –when there are many units and buildings- it takes a longer amount of time to find a proper plan. A HTN planner will attempt to explore all possibilities until it solves the problem or it concludes that there is no solution to a given problem, though this can be fixed by setting a maximum search depth. But of course this will lead to a planner which is less likely to find a solution. Still, because HTN planning is hierarchical the state space is smaller than that of linear planners. Also, the HTN planner needs knowledge of which strategy is best to perform. So an expert is needed to implement sound decisions that the planner will perform. Recently research has been done to enable HTN planners to learn their task without knowledge of the world and even without needing preconditions [ILG06]. So there is a possibility to learn the HTN, but of course the strategy that one would like to use will be lost.

4.3 Available HTN Planners

There is a small amount of HTN planners available. Such as SHOP2, UCMP and EDAPS. SHOP2 is the successor of SHOP, which was a somewhat simpler HTN planner. In contrast to SHOP SHOP2 allows tasks to be partially ordered and SHOP2 incorporates many PDDL features (such as quantifiers and conditional effects) [NAU03]. UCMP is a dated project and does not work properly on current systems. EDAPS is specifically created for design and manufacturing process planning of microwave transmit/receive modules. Then there is Bridge Baron which is designed for playing the bridge card game.

5. TOTAL ANNIHILATION SPRING

As described in chapter 3 there are several open-source RTS games available. For this research Total Annihilation Spring (TA Spring) is selected, since it is very much like commercial RTS games and the development is in an advanced state. TA Spring is not a RTS game on its own, it is an engine in which mods and maps can be loaded. Each mod consists of its own units and buildings. Mods only differ in the unit and building types that can be produced. For this research the mod Xect vs Mynn will be used.

Figure 1: Total Annihilation screenshot

The game play of TA Spring is as follows: at the beginning of the game each player start with a commander, this is a unit which can produce the most basic buildings. These include buildings that provide resources, factories, a radar and defenses. The resources in TA Spring are energy and metal. Energy is provided by buildings that produce energy from solar or wind energy. Metal can be extracted from certain areas by metal extractors. Builders and factories have several building levels. Xect vs Mynn for example has three building levels. These levels can be reached by producing new builders, a level one factory can build a level two builder which can build a level two factory and so on. The higher the build level the more advanced the units and buildings that can be produced. Buildings have a metal and energy cost, units only have metal costs.

6. SHOP2

As HTN planner SHOP2 will be used. This is one of the few available planners which is still supported. There are two versions, a Java and a LISP version. The java version needs to compile problems before they can be fed to the planner which takes too much time to plan in real-time. The LISP version is real-time and therefore this version is used. SHOP2 (Simple Hierarchical Ordered Planner). This is the only planner that is currently supported and is not specifically designed for a certain domain. SHOP2 is written in the Lisp programming language which is logical programming language. Tasks are represented by operators and methods which can be compared with the primitive and non-primitive tasks. An operator consists of the following parts: A *head* which contains the operator name and variables, a *pre-condition expression* which contains the preconditions of this operator, and a *delete-list* and *add-list* which contains the operator's negative and positive effects. Example operators from the SHOP2 example basic-example:

```
(:operator
        ;head
                (!drop ?a)
        ;pre
                ((have ?a))
        ;delete-list
                ((have ?a))
        ;add-list
                ())
```

(:operator (!pickup ?a) () () ((have ?a)))

The operator pickup results in have ?a, the operator drop requires have ?a, when this precondition is true have ?a is deleted.

A method is a prescription of how a compound task is decomposed into primitive tasks. A method consists an head with the methods name and variables followed by a list of preconditions and subtasks in order to accomplish the task. Example of a method called swap:

(:method (swap ?x ?y)

((have ?x))

((!drop ?x) (!pickup ?y))

((have ?y))

((!drop ?y) (!pickup ?x)))))

This method swaps the two variables *?x* and *?y*, when *have ?x* is true *drop ?x* and *pickup ?y* are executed. When *have ?y* is true *drop ?y* and *pickup ?x* is executed. In the case when both *have ?x* and *have ?y* is false this method fails and NIL is returned.

When the HTN is created a problem can be solved, for the swap method the following problem can be solved:

((have apple)) ((swap apple kiwi)))

Note that in this example the pickup operator is always successful, this script allows to hold an infinite number of items .

The first part consists of the state of the world, i.e. the statements that are true. In this case *have apple* is the only thing we know about the world. Next the actions are presented that we want to have executed, which is *swap apple kiwi*. When SHOP2 attempts to solve the problem it tries to execute *swap*, the statement *have apple* is processed in the precondition list of *swap* and will lead to the execution of *drop apple* and *pickup kiwi*. Thus, SHOP2 will return the plan *((!drop apple)(!pickup kiwi))*. When there is no possible sequence of operations to achieve the given goal SHOP2 will return NIL as result.

7. TA SPRING PLANNER DESIGN

As described in Chapter 3 only the strategic level of the RTS game will be dealt with. This will include resource management and the production of buildings and units. Other levels will be left to the already existing AI of TA Spring. There are various strategies available for RTS games, all with their advantages and disadvantages. Well known are Soldiers rush and Knights rush [MOL05]. Soldiers rush is a strategy in which an a player produces cheap units as fast as possible and attacks the opponent in the assumption that the opponent has not already produced a sufficient number of units to stop the attack. Knights is similar but different in the fact that a quick advance in technology is pursued to produce strong units and launch attacks as soon as there is a sufficient number of units. The most common strategy is the balanced build policy, this strategy –as it's name suggests- tries to find a balance in resource management, technological advancement and units production. This strategy is adapted by most human players. For this research also this strategy will be used, since it is most common. The earlier mentioned tactics can be learned easily by both human and computer opponents and they can adapt in next games. This strategy is also the most used in commercial AI for RTS games. In order to make a good comparison with the current AI for TA Spring this is the best option because it is also follows the balanced build policy. When other strategies need to be created, the balanced build balance policy can easily be modified to adapt the needed strategy. To clarify the balanced build policy an example is presented next.

7.1 Build Policy Example

At the start of the game one builder is present (in TA Spring it is called the commander). First the basic buildings are produced, these are a power source, a resource source and a simple factory. Then a radar is built which can detect enemy units, this is an elementary defense mechanism. When the basic factory is ready it can start to build units that can defend the base perimeters. Next a builder of a higher level is produced in order to build more advanced factories. This also creates a need for more resources, so the commander can take care of that by building more power and metal sources. When there are enough units to defend a number of units can be built that can attack the enemy's base. Meanwhile the builders continue to improve the technological advancement. Often a player produces a fast unit that explores the map, this unit is called the scout. Scouting has the following advantages: the player can see enemies coming that are planning to attack his base. The player know the location of the enemy base, so when he decides to attack he knows where to send the units. Even better, if the enemy base is explored directed attack can be launched on certain buildings.

For creating the HTN planner domain a similar strategy is used as described above. It is a combination of the case-based planner described in chapter 3, the build list of the AI that is currently used by one of the AI's which is called NTAI, and the experience of the author with RTS games. Remember that the scope of this research has been limited to building builders, buildings and units. Also note that it is not the goal to create a far superior AI at the first go. The main goal is to create a more efficient AI than the current AI of TA Spring, not an optimal one. The optimal strategy production of buildings and units and resource management is hard to find if not impossible. If the HTN planner turns out to be successful, the planner can be improved to build even more efficient and expended to other tasks in a later stadium. The control of individual units and the collaboration among a group of units are things one can think of. HTN planning has already been applied to collaboration in first person shooters such as Unreal Tournament and F.E.A.R. and turned out to be quite successful [HOA05]. Even though unit controlling in RTS is somewhat different, the cooperation of non-player characters (NPCs) in first person shooters is much more complicated than that of the units in an RTS game.

7.2 Design and Implementation of the HTN Planner

Since SHOP2 and TA Spring are built in different programming languages a means of communication must be developed to enable to exchange data between these two applications. One of the few LISP interpreters that is able to compile and run SHOP2 is Lispworks, there is a free personal edition available which is used for this research. For TA Spring a built-in client is developed that establishes a TCP connection with the server that runs in Lispworks. When the connection is established the client can perform command line command and the output of these commands is returned. Using this setup the client sends the problem to the HTN planner and gets the resulting plan back. This is parsed in to instructions for the builders and factories. The domain needs to be loaded only once since it is static. The HTN planner is consulted whenever a builder or factory has finished its task.

The behavior of the HTN planner is designed as follows. First it is checked whether there is at least one power source and one

energy source. If so the builders and factories that are not doing anything (not busy) will be assigned a new task. At first this will only be the commander, but later on this will be many more builders and factories. Also, a check is done whether there is at least one builder present. If not one is very vulnerable when under attack. Damaged buildings cannot be repaired and destroyed buildings cannot be rebuilt quickly. Even worse, when all factories are destroyed during the attack the player is not able to build anything anymore. When cycling through builders and factories, it is checked which buildings are not built yet and when there are enough metal and energy resources it is built.. This is similar to a build list that is used by the current AI in TA Spring. But there are some advantages using this approach. When a building is destroyed it will not be rebuilt when one makes use of a build list. Also when there are not enough resources to build the next item on the build list it will not be built and the builder or factory will stay idle. In the HTN design the an item is searched which can be built. A danger in this approach is that a builder can resort to building only the cheap buildings in his build tree. This occurs when there are just enough resources to produce the cheapest items in the build tree and the resources are on about the same level when the builder has finished his task. This will be one of the things that are taken into account during evaluation of the planner. The unit with the highest metal cost within the current metal stock is selected for building

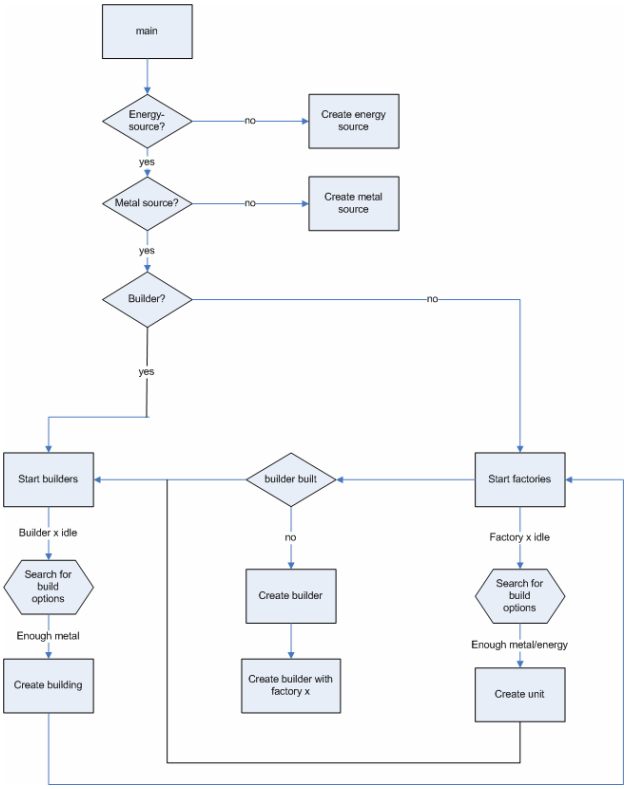

Figure 2: HTN planning tree

When there are not enough resources the current production and consumption are subtracted from each other, if one of these is below a certain threshold a new (energy or metal) source is produced. Using this method sources will be produced only when they are necessary. Whenever a builder is idle because all possible buildings are built or there are not enough resources for building it will be assigned to build a metal source. Metal sources are the most important resource because the building of buildings and especially units depends on the amount of metal in stock. Figure 2 shows the simplified HTN.

In the HTN planner only has four primitive tasks, these are build unit, create-building, spend-energy and spend-metal. Build-unit assigns a task of building a certain unit to a factory. Create-building assigns a task of creating a factory with a certain building. Spend-energy and spend-metal subtract the resources needed from the stored resources in order to have an accurate recording of the resources that are left and can be used by other builders and factories.

8. EVALUATION OF THE PLANNER

Since the planner is designed for playing against other players it will be tested accordingly. The HTN planner is evaluated by running TA Spring 20 times and letting the HTN planner play against the AI that is currently in TA Spring, called NTAI. During these games a number of things will be evaluated. Data is taken from the logs which can be evaluated, a number of assumptions can also be made according to the design of the HTN. A typical game can last between 5 minutes and an hour. Because of these large differences a set time is chosen for each game. Whenever one of the planners is starting to win the evaluation is stopped. Because this situation will result in the one side being destroyed while the other can continue to build. This happens approximately at 12 minutes. This is therefore chosen as the time limit. This will result in one side losing its buildings, units and resources and the other can continue building as normal.

Also to conclude which planner is most successful 10 games will be played all through. From this the number of wins of each planner is taken.

8.1 Data to be measured

In planning in the strategic level there are various aspects that can be measured. These are the results of the planning decisions. For example in resource management the amount of resources and the income and usage of resources can be measured to evaluate its performance. These are measured in regular time intervals of 30 seconds. For strategic level planning the following can be measured.

- Resources, the production of resources in contrast to the consumption.
- Total amount of resources in stock
- Total number of units produced
- Total number of buildings produced
- Number of builders/factories produced

These will be measured during testing. The data will be taken from the log of each game.

When considering low level strategies as well the following aspects can be measured in a RTS game:

- Number of kills
- Number of opponent buildings destroyed
- Number of combat units that exist at a certain time

Note that more is not always better, when a player produces lots of builders most of them will remain idle because there are not enough resources available for producing new buildings.

8.2 Assumptions according to HTN design

There are a couple of basic rules for the planner to be successful and these must be true at all times.

- At least one builder must be in the game, if not it must be produced as soon as possible.

- There must be at least one energy source and one metal source.
- When there is a shortage of resources, new sources must be produced and/or the consumption of resources must be reduced.

Also, a couple of assumptions can be made when using a HTN planner in contrast to using a build list.

- When a building or builder is destroyed the HTN planner will rebuild it after a certain amount of time. The build list planner will just continue building the next item.
- The HTN planner will be more efficient in handling its resources, power and metal sources will only be built when needed.
- The HTN planner should be able to build more units/buildings in a shorter time period due to the more efficient resource handling and the fact that it chooses a build option according to the current amount of resources available.

The planner will be considered successful when these assumptions hold. Also it should be able to outperform NTAI and therefore win the majority of the games played.

9. RESULTS

From the 20 runs the data was taken and to make a proper comparison the average over all the data was calculated. First a general overview of the results is presented then the results from the different data is discussed separately. The tables corresponding to the graphs presented in the next section can be found in Appendix A. The values are rounded to whole numbers because of the large table size and to provide a good overview.

9.1 General results

From the 10 test games which where played all through the HTN planner was able 7 of them. Although this is not significant it is an indication that it outperforms NTAI. This is also showed by the data that is measured which is discussed in the next section. What seems to be the most important flaw in NTAI is that the metal level is very low at all time. Because it also creates many builders which build simultaneously. These use more metal than can be produced, the result is that the metal level stays low.

Both planners create units of equal strength in the beginning of the game. About 8 minutes later the HTN planner starts to build more advanced units. This and the larger number of units results in that the HTN planner's units start to destroy the buildings and units of NTAI.

The assumptions that are made in chapter 8 all hold, when a building or builder is destroyed it does not take long until it is rebuilt. Usually it is reproduced within one minute. This does not completely holds for metal and energy sources. For example, when there is still enough energy when an energy source is destroyed it is not rebuilt. Because it is not needed this is not a problem. The resource management of the HTN planner is more efficient than that of NTAI. NTAI has a very high energy level and a very low metal level. The HTN planner almost never runs out of resources and the consumption is close to the production. This means that there is no over or under production at any time. The HTN planner is also more efficient when it comes to the production of buildings and units. It is able to create more buildings with less builders and more units with an equal number of factories. This will be explained further in the next section. Next the results are discussed in detail. In all figures the green lines represent NTAI and the black lines represent the HTN planner. The x-axis shows the time and the y-axis the measured data.

9.2 Resources

Figure 3 shows the amount of resources at a certain time. The line represents the energy and the dashed line the metal in stock at time t.

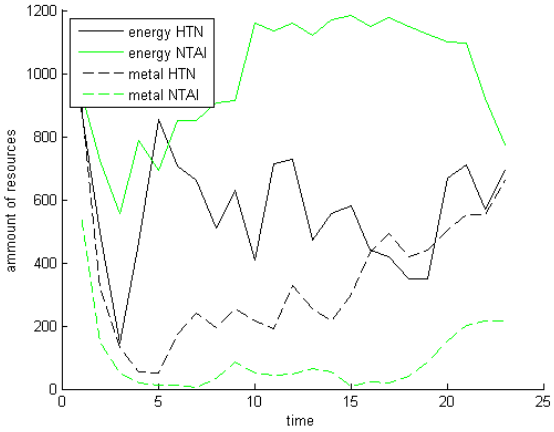

Figure 3: Energy and metal stock

The first thing that is noticed is that NTAI has a very high energy level and a very low metal level. This results in that building is slowed down. NTAI uses multiple builders right from the start which build almost continuously. When the metal is low they have can build as fast as the metal is provided. This is one of the major downsides of NTAI. Also the energy level is always too high, there is more energy than ever will be used. NTAI builds many energy sources very soon in the game. The drop at the end is caused by the destruction of energy sources.

The HTN planner is designed to produce energy sources when the energy level drops below a threshold, this threshold is coupled to the number of factories it has in the game at that time. This explains the fluctuation, whenever the energy level is low a new source is built until it drops below the threshold again. For metal production the same holds except that whenever a builder is idle it is assigned to build a metal resource. This explains the increase in metal during the game.

Figure 4 shows the production and consumption of energy. The line represents the energy production and the dashed line represents the consumption of energy. Again we can see that the NTAI overproduces energy, only in the beginning of the game the consumption is sometimes higher than the production. The drop of the production starting at t=17 is due to the fact that the energy sources are destroyed by the opponent (i.e. the HTN planner).

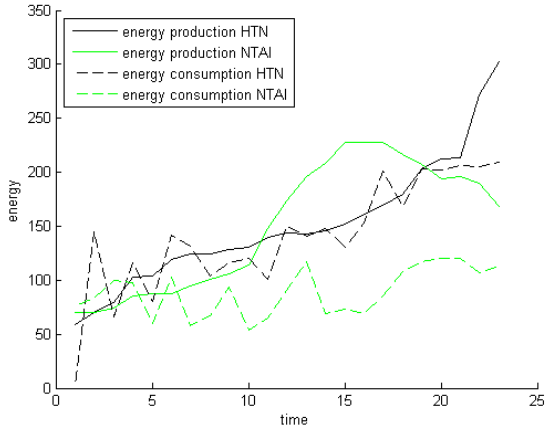

Figure 4: Energy production and consumption

The consumption of energy of the HTN planner fluctuates around the production. Whenever there is not enough energy for building purposes the builder/factory waits until enough resources are available. And when the level drops below a certain threshold new sources are built. The peak at the end is caused by the production of a large power plant that is often produced around that time.

Figure 5 shows the production and consumption of metal, again the line represents production and the dashed line represents consumption.

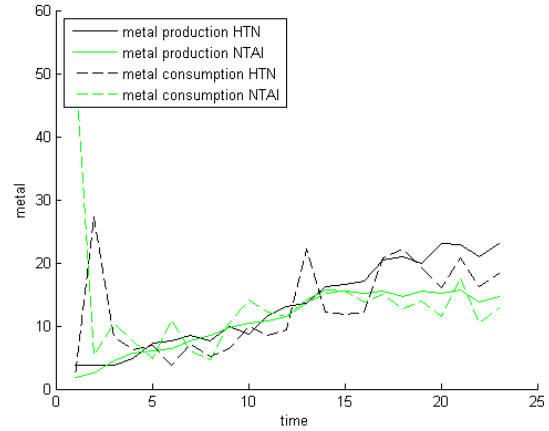

Figure 5: Metal production and consumption

These are very much alike for both NTAI and the HTN planner. But when one takes a closer look it is clear that the consumption of metal of NTAI is always very close to the production. This results in a shortage of metal which can be seen in figure 2. The metal consumption of the HTN planner does not exceed the production very often. When this does occur it is very rapidly dealt with.

9.3 Production

Figure 6 shows the number of builders and factories that are in the game at a certain time. The line represents the number of builders and the dashed line represents the number of factories.

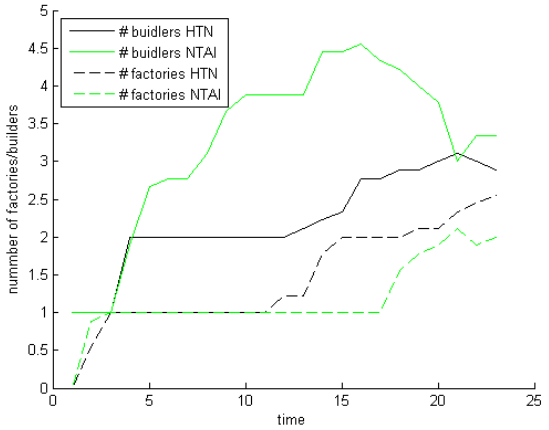

Figure 6: Number of builders and factories

NTAI immediately builds a number of builders, this would be a wise decision if it has enough resources to enable all these builders to build at the same time, it is already shown that this is not the case. At time t=17 the number of builders start to drop again because the opponent destroys them. Striking is that it produces a large number of builders but more factories are produced very late in the game.

In contrast to NTAI the HTN planner does not use many builders. Only when a factory is built a new builder is produced. Also the HTN planner builds an additional factory earlier in the game. This enables the HTN planner to expand more quickly.

Figure 7 shows the total number of buildings and units created. The line represents the number of buildings and the dashed line represents the number of units. Note that this is the sum of buildings and units created, thus when one is destroyed it is not subtracted from the total.

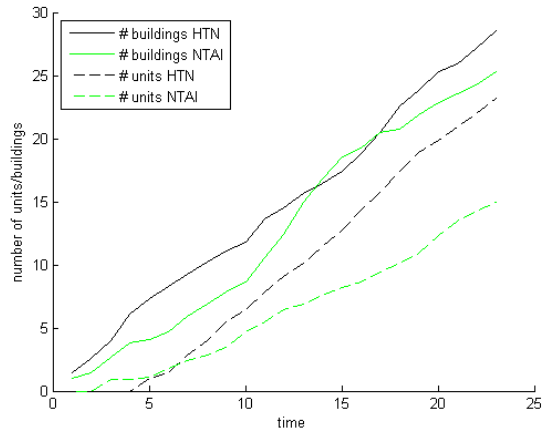

Figure 7: Buildings and units produced

The number of buildings produced does not differ very much between the two planners. But the number of units does. The HTN planner has a higher unit production rate. The additional factory that is created at t=15 by the HTN planner (see figure 6) does not result in a faster building rate. This can be explained: at this time in the game level of metal also starts to increase, then the factories can built more expensive units. But these take more time to build. Therefore the factories do not build more units but stronger units.

10. CONCLUSIONS

In this paper the design, implementation and evaluation of a HTN planner for a RTS game is presented. The implementation is tested on a real RTS game, and is found to be successful. SHOP2 is chosen as planner because of it is able to plan in real-time and is still supported. The evaluation was done in TA Spring because it is open-source and allows an AI to be created separate from the game. The planner is designed according to the 'balanced build policy' which seeks a balance between acquiring resources and producing buildings and units. This is one of the many possible strategies available, in future research other strategies could be used to improve the planner.

The HTN planner performs better than the built in AI of TA Spring. It is able to wins most of the games played. Which is also supported by the results of the measured data, at t=15 the amount of builders and energy starts to drop and the increase in the metal level stalls. It takes some longer time until the factories are destroyed because these are harder to take down. Also resource management and production of buildings and units is proven to be more efficient than NTAI. Especially the resource management is improved, in the current AI the energy level is too high while the metal level is too low. The HTN planner seeks a balance between production and consumption and slowly increases these levels to allow more expensive units and buildings to be produced. Another important fact is that the HTN planner needs less builders to produce at an equal rate.

11. FUTURE WORK

Although the HTN planner has met the goals that have been said. A lot of work is still to be done. In the first place the other levels of the RTS games are to be implemented. The tactical level is also very suitable for HTN planning, attacking in groups can be implemented very well.

But the current planner can be improved as well. The resource management –although good enough- is not optimal, this would require much more research. Also the placement of the buildings is to be improved. In the current AI it just searches for space in the map without making tactical considerations. The current planner only produces buildings with defense purposes once. These buildings can provide excellent defense for a group of buildings.

In a later step the planner could be evaluated against experienced human players. Since human players are still able to defeat any AI built so far for RTS games this might be a good test bed.

ACKNOWLEDGMENTS

The author likes to thank Mannes Poel for his feedback and patience.

REFERENCES

[ACM] ACM Special Interest Groups Proceedings Template: http://www.acm.org/sigs/pubs/proceed/pubform.doc

[AHA05] Aha, D W., Molineaux, M., Ponsen, M.J.V., Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game, *International Conference on Case-Based Reasoning*, 5-20, 2005

[BUR03] Buro, M. Real-Time Strategy Games: A New AI Research Challenge. *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 9-15, 2003.

[BUR04] Buro, M. Call for AI Research in RTS Games, *AAAI-04 AI in Games Workshop*, 139-142, 2004

[CHE04] Cheng, D.C., Thawonmas, R.. Case-based plan recognition for real-time strategy games. *Proceedings* of the Fifth Game-On International Conference (pp. 36-40), 2004.

[CHU05] Chung, M. Buro, M. and Schaeffer, J. Monte carlo planning in RTS games, *Proceedings of IEEE Symposium on Computational Intelligence and Games*, 117–124, 2005.

[ERO94] Erol, K., Hendler, J., and Nau, D., UCMP: A Sound and Complete Procedure for Hierarchical Task-Network Planning, *Proc. of the Second nternational Conference on AI Planning Systems (AIPS-94)*, 249-254, 1994.

[HOA05] Hoang, H. Urban, Stephen L. , Avila, Hector M., Hierarchical Plan Representations for Encoding Strategic Game AI, *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference*, 2005.

[ILG06] Ilghami, O. Nau, D.S., Munoz-Avila,H., Learning to Do HTN Planning, *Proceedings of the Sixteenth International Conference on AI Planning and Scheduling*, 2006.

[MOL05] Molineaux, M., Aha, D.W. and Ponsen, M.J.V., Defeating Novel Opponents in a Real-Time Strategy Game, *Proceedsings of Reasoning Representation, and Learning in Computer Games. Papers from the IJCAI Workshop*, 2005.

[MUN04] Munoz-Avila, H. and Fisher, T. Strategic Planning for Unreal Tournament Bots. *In AAAI Workshop on Challenges in Game AI*, 2004.

[PON05] Ponsen, M. J.V., Muñoz-Avila H., Spronck, P. Aha, D. W.. Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. *AAAI*, 1535-1540, 2005.

[NAU03] Nau, D. et al., SHOP2: An HTN Planning System, *J. Artificial Intelligence Research*, 379–404, Dec. 2003.

[SMI98] Smith, S. J. J., Nau, D. S. ,Throop, T. A.. Computer Bridge - A Big Win for AI Planning, AI Magazine 19, 93-106, 1998.

Appendix A: Planner results rounded to whole numbers.

Table 1: HTN planner results

energy	energy production	energy consumption	metal	metal production	metal consumption	# builders	# factories	# buildings	# units
890	59	3	915	4	2	1	0	1	0
496	70	145	322	4	27	1	1	3	0
144	80	66	130	4	8	1	1	4	0
466	103	116	56	5	6	2	1	6	0
856	104	81	52	7	7	2	1	7	1
708	119	142	172	8	4	2	1	8	1
660	124	132	240	8	7	2	1	9	3
510	125	103	196	8	5	2	1	10	4
628	129	116	253	10	6	2	1	11	5
409	131	120	215	9	10	2	1	12	6
715	140	101	191	12	9	2	1	14	8
728	144	150	329	13	9	2	1	15	9
471	143	141	256	14	22	2	1	16	10
557	145	148	215	16	12	2	2	16	11
581	152	131	301	17	12	2	2	17	13
440	161	153	432	17	12	3	2	19	14
419	170	201	494	20	21	3	2	20	16
348	179	169	418	21	22	3	2	23	17
349	203	203	441	20	19	3	2	24	19
668	212	202	504	23	16	3	2	25	20
710	213	207	553	23	21	3	2	26	21
570	272	205	552	21	16	3	2	27	22
692	302	210	663	23	18	3	3	29	23

Table 2: NTAI results

energy	energy production	energy consumption	metal	metal production	metal consumption	# builders	# factories	# buildings	# units
941	70	77	547	2	52	1	0	1	0
724	70	83	150	3	5	1	1	1	0
555	74	99	52	5	10	1	1	3	1
786	86	98	19	6	8	2	1	4	1
693	88	60	13	6	5	3	1	4	1
851	88	103	13	6	11	3	1	5	2
850	94	58	4	8	6	3	1	6	2
906	101	67	35	8	5	3	1	7	3
915	106	93	86	10	10	4	1	8	3
1161	114	54	49	10	14	4	1	9	5
1136	148	65	43	11	12	4	1	11	5
1160	173	91	48	12	12	4	1	12	6
1120	196	117	66	14	14	4	1	15	7
1169	208	69	56	16	15	4	1	17	8
1184	228	73	8	16	15	4	1	19	8
1150	228	69	24	15	14	5	1	19	9
1176	228	85	18	16	15	4	1	20	9
1149	217	107	40	15	13	4	2	21	10
1125	207	117	87	16	14	4	2	22	11
1102	194	120	152	15	11	4	2	23	12
1096	196	120	201	16	18	3	2	24	13
922	190	107	216	14	10	3	2	24	14
773	168	113	217	15	13	3	2	25	15