Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

AI for Massive Multiplayer Online Strategy Games

Alexandre Barata, Pedro A. Santos, and Rui Prada
Instituto Superior Tecnico, Technical University of Lisbon, INESC-ID, Lisboa, Portugal.

Abstract

Massive Multiplayer Online Strategy games present several unique challenges to players and designers. There is the need to constantly adapt to changes in the game itself and, sometimes, the need to achieve a certain level of simulation and realism, which typically implies battles involving combat with several distinct armies, combat phases and diferent terrains; resource management which involves buying and selling goods and combining lots of diferent kinds of resources to fund the player's nation and cutthroat diplomacy which dictates the pace of the game. However, these constant changes and simulation mechanisms make a game harder to play, increasing the amount of effort required to play it properly. As some of these games take months to be played, players who become inactive have a negative impact on the game. This work pretends to demonstrate how to create versatile agents for playing Massive Multiplayer Online Turn Based Strategy Games, while keeping close attention to their playing performance. In a test to measure this performance the results showed similar survival performance between humans and AIs.

Introduction

Game AI can take several forms and purposes. Sometimes game AI's purpose is purely to advise the player (e.g. SimCity 4's advisors), sometimes its purpose is to allow the players to focus on high-level strategy by automating lower-level tasks (e.g. game pathfinding), and sometimes its purpose is to challenge players (the AI-controlled rival nations in the Civilization™series). Good AI is particularly important in single player games, where it must provide adequate challenges to the player all by itself. However, even in multiplayer games, AI can play a major role in solving specific problems as this work will try to show.

The main goal of this work was to create AI capable of playing in the complex Massive Multiplayer Online Turn-Based Strategy Game (henceforth referred to as MMOTBSG) world of Almansur. As a MMO, Almansur is a game in constant change, and like many other games of its specific genre, it has a problem with inactive players. Before the AI player was implemented, these were treated exactly like active players. Due to Almansur's conquest based gameplay, however, active players near inactive players could easily take their territories without a fight, earning an unfair advantage for the rest of the game.

This problem results from MMOTBSG matches taking a very long time to be completed (typically more than 1 month) and is especially hard to solve in games with realistic space representation (**?**) worlds such as Almansur. Thus, an essential objective of this work is having the created AI play better than an inactive player.

Due to Almansur's design as a historical simulation game, removing an inactive player from a match is not possible as that would imply removing an active part of the world from the match and ruin the intended realism. Matches that can take days or months to complete are very likely to have some players going inactive as their time available to play changes. Although these players are no longer playing the game, their territories and resources are still represented in the world and can be plundered at will.

The creation of an AI player to replace inactive players could also allow Almansur to be played in single player mode, and allow mixed matches where several humans and AIs compete. To accomplish this work's objectives, using artificial agents was a very interesting prospect: it is natural to classify players as game agents (**?**). Thus, the key capability needed by these agents in order to succeed is the ability to plan and execute short and long term game strategies and tatics in a dynamic way that can adapt to constant changes in the game itself.

Related Work

To the authors' knowledge, there is a lack of AI developed for MMOTBSGs. This is due to a few factors:

- MMOTBSGs tend to be resource hogs, with severe scalability issues due to the massive amount of players per game, which tends to limit the amount of players that can play at the same time. If there is no space for all the humans who want to play, why even consider AI?
- MMOTBSGs tend to emphasize interaction between the players, with resource trading, teamwork and communication being major factors, which would make the required AI either too complex or ineffective at its job.

- Inactive players are generally not too much of an issue in most MMOTBSGs (because most do not have realistic space representation), they are a part of the game instead and help newer players in their early play while not being major factors in the mid/late game.

Thus, due to lack of previous related work, the authors have instead focused on analyzing a non-MMO TBSG with good AI: Civilization IV (abbreviated CivIV). Unlike Almansur, however, CivIV focuses on the single player facet of the game with a secondary multiplayer component for up to eighteen players. Single player is possible due to Civ's AI opponents. The AI can also help the player manage low level tasks by assigning workers to the best available tiles for example and several advisors for each facet of the game are present. Civ IV's AI is soft-coded, uses fuzzy logic (**?**) and cheats in order to achieve its ultimate goal: to be fun (**?**).

The inactive player problem is easily solved in CivIV by using the AI. Whenever a player leaves the game the other players are prompted to vote over whether to wait for him to return, to save the game until that player returns or to simply replace him with an AI player. Players joining a game can take over previously AI-controlled nations as well. CivIV players can also focus on high level tasks by delegating low level tasks to the AI.

Although CivIV's AI has solved a lot of problems for the game, it cannot be easily adapted to Almansur as CivIV is not a MMO game in constant evolution, and thus its developers scripted much of the race/resource/facility specific decisions, making the hard task of adapting a game AI to another game even harder (**?**).

As these factors are in constant change in Almansur (new races/resources/facilities can be added or removed at any time), the AI is required to not rely on scripting and rather do a more versatile approach able to adapt on the fly. It was also the desire of this paper's authors to have a non-cheating AI, which invalidates CivIV's approach to difficulty levels where harder difficulties use cheating AI.

Almansur

Almansur is a browser MMOTBSG in a medieval-like world that combines empire-building, turn-based gameplay with the defining MMOG characteristic of allowing hundreds of players to play simultaneously (**?**). In a single game there are many independent *Land*s of different races, each controlled by a single player.

Victory comes through the completion of several different game objectives. In order to achieve these objectives, the players improve their territories by building / upgrading facilities on them, by recruiting / training armies to defend their *Land* and attack others, and by creating alliances between themselves through diplomacy.

In order to understand how to *implement an AI player capable of playing MMOTBSGs*, the authors have identified several independent game facets in this genre, and in Almansur in particular:

- *Economy :* The economy facet is a part of all strategy games where players need to build their own facilities in order to receive resources and create armies. The more units/facilities/resources/technologies there are in a game, the more complex its economic system usually is.
- *Military :* Last but not least, Military exists in any strategy game where there is the notion of armies and their composing units. Different strategy games have completely different styles of military control that a player must master in order to win. Even so, the path to victory is very influenced by military control and as such this is perhaps the hardest facet to master in such games.

Playing Almansur implies playing all the facets described above at the same time and while they can be treated as mostly independant entities (**?**), all affect the performance of the player in a very significant, unique fashion. Almansur's environment can be classified (**?**; **?**) as *inaccessible*, *deterministic*, *dynamic* and (partially) *discrete*, which makes it a very complex environment to design AI for.

Furthermore, the nature of the interaction between the AI and its environment is quite complex as well, as the AI has the same knowledge access as human players, using the exact same interface as these to access and modify the game. Finally, the AI needs to be able to adapt itself to modifications in the game without requiring code changes, as it represents an independent system.

Given the points presented above, the objective of solving the inactive player problem and the need to somehow simplify the huge task that is playing Almansur in all of its facets while keeping the AI as dynamic as possible, it was concluded that the AI needed two main competences in order to succeed in this environment, which are presented below:

- *Strategic Competence* - Ability to do long-term strategic thinking, for example "What goals should I create for the current turn?".
- *Tactical Competence* - Ability to do short-term strategic thinking, for example "How to complete all the conquer goals for the turn with my current armies?".

Agent Architecture

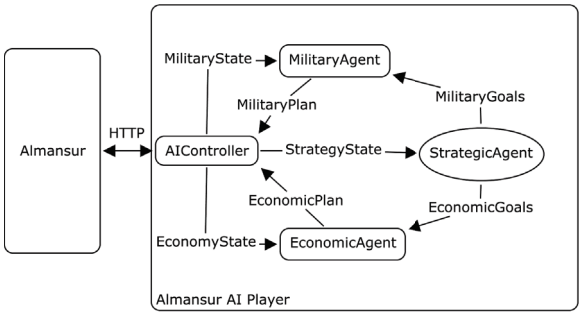

Figure 1: The AI Player's Architecture

As seen in Figure **??**, the AI was implemented by three distinct agents: the strategy, economy and military agents. To interact with the game engine, a specific module named *AIController* was created. This module is responsible for

collecting the information the agents will need to plan, which is stored in their specific state, and for converting agent's plans into game actions. In order for each agent to play, while being as dynamic and resilient to game changes as possible, a custom search-based planning system (**?**) was created, which uses agent-specific goals and operators as well as the state information to plan out the best perceived course of action for the turn.

Strategy Agent

The strategy agent is responsible for the AI's high level strategic decisions, which take the form of goals for the other agents (**?**). This agent's state contains information about the Land's map territories, about the Land's resource production levels and about whether the AI's Land is at war with other players or not and uses this information to decide on which goals to set for the Military and Economic agents. Economic goals can be of two kinds: resource goals and production goals. Resource goals are achieved when the Land stockpiles a certain amount of resources of a kind while production goals require the Land to stockpile a certain amount of production per turn of the desired resource.

Military goals can be of three kinds: army size goals, facility goals and conquer goals. Army size goals are achieved when the Land's armies cost a certain percentage of the Land's income to maintain and the Land's fortresses have a certain amount of troops guarding them, facility goals are achieved when the Land upgrades its secure (in a territory with fortress) military facilities up to a certain level and conquer goals are achieved when the Land conquers a certain territory.

The strategy agent was implemented by a simple decision tree following interviews with expert human players. When at war, production goals are cancelled, resource goals for resources required to recruit troops are created / increased, army size and facility goals are increased and conquer goals prefer to target enemy territories, if possible. When at peace production goals are set to increase the overall production of the Land with focus on the most important resources, resource goals are set in order to ensure there are always reserve resources available to upgrade facilities / recruit in case of sudden war, army size and facility goals are decreased and conquer goals only target uncontrolled territories.

Economy Agent

The economy agent is responsible for solving the economic goals. This agent's state contains the AI's Land economic details: how much resources the Land has, which resource-producing facilities exist and how much they produce, the income of each resource, the upkeep of each resource, how many resources of each kind are avaliable on the market and their price.

Using this information and the economic goals for the turn this agent is able to determine how to best satisfy them given the Land's current economic situation.

The initial node of the economic planning system is the state of the current turn. To create successors for the search, the AI applies several operators to this state: tries to upgrade every resource-producing facility the Land has, tries to buy all market resources available, tries to sell all market resources avaliable and tries to pass the current turn (and thus, predict how the game will change into the next turn, in the economic view of resource production and consumption). The *pass turn* operator is especially interesting, as it allows the AI to understand the result of its actions over time before it commits to them.

Since all of the operators described above have arguments, some of which with continuous ranges such as the market related ones, the total amount of possible operators that can be applied each turn is huge. Considering that the pass turn operator allows all other operators to be used again, in the context of the new turn, the total search space is too large to search blindly. Therefore, in order to reach a compromise between playing quality and software performance, finding a way to prune the search graph and guide the planner's search was a requirement.

The market operators' arguments were restricted to very limited ranges: instead of trying to trade many different amounts of resources at the market, the AI tries to trade as much resources in a single operation as possible given the resources available both in the market and on the Land. In order to guide the search, a system based on both the concept of heuristic (estimated distance to a goal from the current state) and cost to get to the current state so far was implemented:

The heuristic function used by this agent to evaluate how far a certain state s is from a certain goal is slightly different for production goals pG and resource goals rG. The production goal heuristic formula is:

$$h(pG,s) = \sum_{\text{r in pG}} (oP(r) - fP(r)) * mP(s,r) \qquad (1)$$

- r represents the *resource* for which the agent has a production goal pG.
- $oP(r)$ is the objective production amount of *resource* set by the production goal pG.
- $fP(r)$ is state s's future production of *resource* - that is, the production the AI's Land will have when all upgrading facilities complete.
- $mP(s,r)$ is state s's market price of *resource* for the AI's Land.

The resource goal heuristic formula is:

$$h(rG,s) = \sum_{\text{r in rG}} \frac{oR(r) - cR(r)}{fP(r)} * mP(s,r) \qquad (2)$$

- r represents the *resource* for which the agent has a resource goal rG.
- $oR(r)$ is the objective *resource* amount set by the resource goal rG.
- $cR(r)$ is the current amount of *resource* for the AI's Land.

Using these heuristic functions, nodes with smaller heuristic values are preferred. One important thing to note which is not represented in the equations is that if the result

of $oP(r) - fP(r)$ or $oR(r) - cR(r)$ is negative, it is set it to 0 instead in order to avoid giving value to overcompleting a goal, which was proven by testing to cause the planner to prefer overcompleting single goals rather than completing all goals equally.

As cost, the turn number works well since faster is better in Almansur when all goals are achieved:

$$c(s) = t(s) \tag{3}$$

- *t(s)* is state *s*'s turn number.

These two values work well together for guiding the economic agent to its current goals. However, there are typically many different ways to reach these goals, resulting in different final states - a final state might be reached which has more overall resources than another, yet both reached the proposed goals on the same turn, for instance. In these situations, we use the state evaluation function *sEval* described below to decide what state is, overall, the best:

$$sEval(g,s) = (tA(g,s) - iP * \delta(turn)) * 0.98^{\delta(turn)} \tag{4}$$

- *tA(g,s)* represents the *totalAssets* of the Land for the state's turn, which are given by:

$$tA(g,s) = \sum_{resource} (a(s) + pA(s)) * mP(s) * gC(g) \tag{5}$$

  - *resource* represents each of the resource types available to the AI's Land.
  - *a(s)* is the amount of *resource* in the Land's coffers for the state's turn.
  - *pA(s)* is the amount of *resource* produced every turn by the Land for the state's turn.
  - *mP(s)* is the state's market price of *resource* for the AI's Land
  - *gC(g)* represents how much the Land currently requires a particular resource type. These values depend on the Land's goals: if there is a goal for *resource* its value is 5, if there is no goal for *resource* its value is 1. These values were throughly tested in order to ensure that goal resources have priority, but not so much priority that all other resources would be recklessly sacrificed in order to obtain them. Having a bigger range of values for this variable would be useful in order to create a better balance when there are several competing resource goals with different priorities but this wasn't implemented due to lack of time to properly test its implications.

- *iP* represents the *initialProduction* of the Land, which represents the *productionAssets* of the turn where economic planning started.
- δ*(turn)* represents the amount of turns which have passed in the current plan state since the economic planning started.

$0.98^{\delta(turn)}$ represents the penalty awarded to the *sEval* value for passing a turn, which favours exploring the nodes which achieve better results in the least amount of turns.

When completing all economic goals takes too long (more than 2 seconds), the planner returns the best plan found during the search. In nearly 100% of the cases this is a plan with actions that extend over the current turn, and thus effectively a complete plan for the current turn already. Each turn the economic agent creates a new plan, as in most cases the economic situation changes significantivelly as territories are lost / conquered and it would be more complicated to adjust the old plan than to create a new one, based on the current turn.

Military Agent

The military agent is responsible for the details of war. This agent's state contains the AI's Land military details: which armies are on the map, where they are, what orders they currently have, how much experience they have, what kind of troops they have, how much status (army characteristic that determines how rested it currently is) they have, which military facilities exist, the map itself and by which types of territories it is composed, how many resources are available for military use, how many resources the Land is paying in upkeep to supply its armies, how many of these resources are being produced and how many days and turns (a turn is typically 30 days) have passed since the start of the game.

This information is then compilated into influence maps which give a quick overview of the field when decisions are required (**?**). Using the influence maps and the strategy agent's goals for the turn this agent is able to determine how to best satisfy them given the game's observable military situation.

The initial node of the military planning system is the state of the current turn, similar to how the economic agent works. However, in the military agent's case, the huge search space problem is solved by three sub-agents whose decisions combined make for a complete military plan:

- The *Military Facility Agent* is responsible for building a military infrastructure and, depending on whether the AI is at war with other players or not, to upgrade this infrastructure to the required levels for optimal army recruitment options. The military facilities are only built / upgraded at territories with a fortress in them in order to ensure their safety from enemy attacks. The decision process for upgrading the military facilities is simple and does not involve search: if the ironworks is below the required level and can be upgraded (resources are available and the facility is not in the process of being upgraded already), upgrade it, then do the same for the recruitment center.
- The *Military Recruitment Agent* is responsible for using the AI's resources to recruit military units, creating balanced, strong armies to attack the AI's enemies and dynamically filling Fortress' garrisons using the influence maps to guide whether there is danger nearby and, thus, more resources should be devoted to defense or no danger is present and a token garrison is enough. Armies are preferably recruited at the territory with best military facilities, in order to make these armies as efficient as possible. Recruitment requires resources, proportional to which troops and how many are being recruited. This means that, like the decision to trade at the market on the economic agent, there are nearly infinite combinations of army compositions and amounts possible. Thus, as the

game is designed to favour balanced armies, the authors opted for creating army templates, and to recruit the maximum amount of troops that resources allow, as long as this recruitment doesn't take the Land's upkeep above its production. As armies are not worth much without training, they are kept at training order after recruitment, until they are experienced enough to be used in combat.

- Finally, the *Military Command Agent* is responsible for directly controlling the existing trained armies. In order to determine which armies should conquer which territories and on what order, a scheduling system was implemented which roughly estimates how much time - in game days, not turns, as most armies can conquer several territories on a single turn - a given army will take to accomplish a particular order. In this system, to determine how long it will take to conquer a territory simple pathfinding (which takes the characteristics of the terrain into account) is used to estimate the amount of time the army will need to reach that territory using standard speed and then estimate the amount of time the army will need to actually conquer the territory using a simple formula available on the game's wiki. The military planner uses two operators: one that represents sending an army to conquer a certain territory, which creates a scheduled event which keeps that army from being able to be sent anywhere else while conquering that territory and one operator which, much like the pass turn operator on the economic planner, advances time up to the closest scheduled event creating a new state where at least one army is able to move again (as it has completed its previous orders) and updating territory control, since territories will change ownership as they are conquered. A problem with this system, however, is that the larger the amount of armies that must be managed at the same time, the more time this planner's search takes. Due to way the game works, larger armies conquer things faster than smaller ones, up to the point where having armies smaller than a certain amount is simply useless when trying to conquer territories as they take too long to do it. Thus, to solve this problem two very simple solutions were implemented, based on expert play:
  - Armies which start a turn in the same territory are merged into a single, more powerful, combined army.
  - Armies with less than a critical mass amount of troops are sent back to the AI Land's main recruitment territory in order to be merged with armies who are training there.

The military agent was split into these three sub-agents in order to both simplify the military agent's planning, which would be quite complex otherwise and because it was a decision which made sense, as each of these three sub-tasks are mostly independent from each other and, thus, could easily be split without harming the AI's play quality.

Results

The previous section described how each of the essential tasks to play in a MMOTBSG game was implemented in the Almansur AI by means of specialized agents. All of the agents combined are able to play the economic and military sides of Almansur. In this section empirical results are provided that show that the AI was, indeed, successful at providing a suitable replacement for inactive players, while also being able to play a full Almansur game.

The AI was tested on a historical scenario based on the map of France in the 12th century. There were 15 human players who had previous experience in the game and 17 AI-controlled players. The game took a month to complete, with one turn processing every day except weekends for a total of 30 turns. While conducting the tests, the following results for AI / Human territory control were obtained:

Turn	AIs	Humans
0	7.53	8.33
11	8.00	7.80
13	8.05	7.73
18	7.53	8.33
30	7.41	8.47

Table 1: Average AI / Human territories by turn

What the results in this table point out is that, around turn 11/13 at least, AI players are actually controlling more territories on average than the human players. However, as the game progresses the human players recovered by having better coordination and military control of their armies than AI's.

When deciding which production facilities to upgrade, expert human players consider their cost-effectiveness, that is, the amount of resources and time it takes to upgrade it compared to the amount of extra resources produced. The AI does the same, upgrading facilities with higher suitability further than it upgrades facilities with low suitability, while not neglecting to upgrade these when the cost of upgrading the best suitability facilities becomes high enough.

Also the AI, like a human player, comes to the help of its allies when they are attacked, as alliance war declares war for all of its members, thus creating an enemy for the AI player. In case that is the closest enemy to the AI, its armies will be moved to its ally's battlefield, helping out in the war. In case the AI was already being attacked, it will try to defend itself and counter-attack the closest enemy first, which makes sense from expert human perspective.

Finally, the following results for AI/Human player final victory points (abbreviated VPs) were also calculated (the initial VPs for each Land was subtracted from the end VPs as each Land starts with a different amount):

	AIs	Humans
VPs	711797	773809
σ	348651	645071

Table 2: Average AI / Human VPs for the game and its σ

As the above table shows, the average VPs of AI's and Humans was close, with the humans having a slight lead. It also also very interesting to note that the variance σ of human players was much higher than that of AI players. The

reason this happened was because, while the test game was played by experienced human players, some went inactive for several turns, thus leading to very different performances between the human players, while all AI's played the same way, with the only difference being how the other players interacted with them and their positions in the map. Thus, one conclusion from these statistics is that the AI is capable of holding its own against experienced players.

Conclusion and Future Work

This paper presented the implementation of a MMOTBSG playing AI, able to replace an inactive player at any moment's notice (it should be noted that the details of when/why a player is deemed inactive and replaced by the AI are beyond the scope of this work) and play a full game.

The reason this was possible was because the authors managed to identify and separate different aspects of the game, which when considered all at once would create impossibly large search graphs, yet can be solved individually in a relatively short time, leading to local solutions to particular problems.

The AI architecture using multiple agents also has the advantage of being easily expanded and modified, as each agent is easily modified on its own, without any risk of changes affecting other agents. Still, there were several hard decisions to make while creating the AI system presented in this paper.

One of them was deciding whether to separate the market planning from the facility planning. Doing so would speed up both processes considerably, however it was eventually decided to keep them together as there were many situations where buying / selling resources at the market was a prerequisite for upgrading facilities due to lack of key resources, and the option of aiming for predetermined amounts of resources when market trading was considered too static. There are many other things which can be improved in the future such as:

- *Improving army control* in the Military Agent's Command sub-agent by adding different army profiles such as distinguishing between infantry and cavalry and different kinds of armies such as melee heavy or ranged heavy, which would help determine better army compositions depending on the existing battlefields.
- *Expand the Strategy Agent's decision tree* by adding war preparation to the existing peace and war nodes, which would prepare the AI's Land for war and declare it after a few turns, which would help make the transition from peace to war smoother and allow the AI to take the iniciative rather than simply reacting when attacked.
- *More widespread influence map usage* in all agents, rather than only the Military Agent, which could be useful in order to avoid having the Economy Agent building important facilities in dangerous territories and help the Strategy Agent decide which territories to focus on each turn.

A future application of the AI would be game scenario balancing, as the AI can play a very large amount of turns in a short time, and all AI controlled Lands play at the same skill level. These two characteristics allow for extensive, yet inexpensive, scenario testing where the major factors of Land success are imbalances in the map itself.

It would also be interesting to create AI advisors in the future based on this work, supporting players that wish to be helped with strategies and tactics. Finally, it should be noted that the version of Almansur that includes this paper's AI system will be launched very soon, allowing for extensive testing with a vast amount of players.

Acknowledgments

This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

AlmansurLDA. 2007. Almansur. URL: http://www.almansur.net/.

Champandard, A. 2003. *AI Game Development*. New Riders.

Dill, K., and Papp, D. 2005. A Goal-Based Architecture for Opposing Player AI. Artificial Intelligence and Interactive Digital Entertainment Conference.

Johnson, S. 2008. Playing to lose: AI and civilization. Game Developer Conference.

Jones, J.; Parnin, C.; Sinharoy, A.; Rugaber, S.; and Goel, A. 2009. Adapting game-playing agents to game requirements. Proceedings of the Fifth Artificial Intelligence for Interactive Digital Entertainment Conference.

MCCoy, J., and Mateas, M. 2008. An Integrated Agent for Playing Real-Time Strategy Games. Twenty-Third AAAI Conference on Artificial Intelligence.

Pottinger, D. 2000. Terrain Analysis in Realtime Strategy Games. Computer Game Developer Conference.

Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach - Second Edition*. Prentice Hall.

Santos, P. 2007. MMOSG gameplay characteristics. URL: http://thoughtsongames.blogspot.com/2007/07/mmosgs-gameplay-characteristics-part-2.html.

Wooldridge, M. 2009. *An Introduction to MultiAgent Systems - Second Edition*. John Wiley & Sons.

Wray, R.; Lent, M.; Beard, J.; and Brobst, P. 2005. The design space of control options for AIs in computer games. Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games.