Real-Time Strategy Gaines: A New AI Research Challenge

Michael Buro
Department of Computing Science, University of Alberta, Edmonton, AB, T6J 2E8, Canada
email: mburo@cs.ualberta.ca

Abstract

This poster motivates AI research in the area of real-time strategy (RTS) games and describes the current status of a project whose goals are to implement an RTS game programming environment and to build AIs that eventually can outperform human experts in this challenging and popular domain.

1 Real-Time Strategy Games

Commercial computer games are a growing part of the entertainment industry and simulations are a critical aspect of modern military training. The two fields have much in common, cross-fertilize, and are driving real-time AI research [Herz and Macedonia, 2002]. With the advent of fast personal computers, simulation-based games have become very popular. Today, these games constitute a multi-billion dollar enterprise. Examples are sports games and real-time strategy games. The common elements of simulation games are severe time constraints and a strong demand of real-time AI which must be capable of solving real-world decision tasks quickly and satisfactorily. Popular simulation games are therefore ideal test applications for real-time AI research.

Real-Time-Strategy (RTS) games - such as the million-sellers Starcraft by Blizzard Entertainment and Age of Empires by Ensemble Studios - can be viewed as simplified military simulations. Several players struggle over resources scattered over a 2D terrain by setting up an economy, building armies, and guiding them into battle in real-time. RTS games offer a large variety of fundamental AI research problems, unlike other game genres studied by the AI community so far:

• Resource management. Players start off by gathering local resources to build up defenses and attack forces, to upgrade weaponry, and to climb up the technology tree. Proper resource management is a vital part of any successful strategy.

• Decision making under uncertainty. Initially, players are not aware of the enemies' base locations and intentions. They have to gather intelligence by sending out scouts. If no information is available yet, the players must form plausible hypotheses and act accordingly.

• Spatial and temporal reasoning. Static and dynamic terrain analysis as well as understanding temporal relations of actions is of utmost importance in RTS games - and yet, current game AIs largely ignore these issues and fall victim to simple common-sense reasoning [Forbus *et al.,* 2002].

• Collaboration. In RTS games groups of players can join forces and intelligence. How to coordinate actions effectively by communication among the parties is a challenging research problem.

• Opponent modeling, Learning. One of the biggest shortcomings of most (RTS) game AI systems is their inability to learn from experience. Human players only need a couple of games to spot opponents' weaknesses and to exploit them in upcoming games. Current machine learning approaches in this area are inadequate.

• Adversarial real-time planning. In fine-grained simulations, agents cannot afford to think in terms of micro actions. Instead, abstractions have to be found which allow a machine to conduct forward searches in a manageable abstract space and to translate found solutions back. Because the environment is also dynamic, hostile, and smart - adversarial real-time planning approaches need to be investigated.

Playing RTS games is challenging. Even more challenging is the creation of autonomous real-time systems capable of outperforming human experts in this domain. Because search space abstraction, real-time planning, and temporal and spatial reasoning are central to many other problems, the scope of applications seems endless. One example is high-performance combat simulators which are in large demand for training military personnel today and will become the core of automated battlefield decision-support systems of tomorrow, [von der Lippe et al., 1999] predicts that 20% of the US armed forces will be robotic by 2015.

2 An RTS Game Programming Environment

The lack of AI interfaces even in upcoming RTS game titles makes it hard to conduct real-time AI research in this area and to compare the strength of the resulting AI systems with that of human experts. In order to solve this problem we launched an open source RTS game programming project [Buro, 2002] with the following goals:

• Building a hack-free server-client RTS game system. At the core of the system is a simulator to which players connect via UNIX sockets (Fig. 1). The unique system features include: server-side simulation - which only sends visible information to clients thereby rendering common map-revealing client hacks useless - and an open message protocol that allows AI researchers and players to connect *whatever* client software they like.

• Sparking competition among players and researchers. Popular games in which human players still have the upper hand are ideal test-domains for AI research. Unlike the confined GUIs of commercial RTS games, our open design allows the construction of hybrid AI systems in which the human general is aided by AI modules of growing capabilities. Competitive game playing on an open Internet RTS game server is therefore likely to improve AI performance and ergonomic GUI design.

• Applying planning and machine learning techniques to RTS games. Classic game AI methods - such as alpha-

beta search - cannot be applied directly in this domain due to the large number of game objects, imperfect information, and real-time demands. Search space abstraction, hierarchical planning, and machine learning arc approaches to overcome these problems.

Our initial software release implemented a basic RTS programming infrastructure . Recently, we added terrain features (water, ground, plateaus, and ramps), view obstruction by terrain, and a split-view GUI which allows to watch different parts of the playing field simultaneously. In order to maintain the high simulation speed, terrain and vision are based on tiles (Fig. 2). Soon, the complete source code of our RTS programming environment will be released. It is being used in two thesis projects dealing with machine learning of low-level unit behavior and abstraction, planning, and heuristic search in RTS games. We hope to report first results in the RTS game AI domain later this year and to spark enough interest to be able to organize man-machine and machine-machine RTS game tournaments in the near future. For this it will be necessary to define standard game setups including unit properties (size,speed,sight and attack ranges,...), technology tree, and resources. In contrast to commercial products, the focus here will be on simplicity to ease the initial AI development.

3 Related Work

The growing literature on AI in commercial games surely indicates the demand of smarter game AI. Many articles on planning are relevant to this project. Due to space limitation we only briefly mention J. Laird's Soar architecture and its application to first-person shooter games [Laird, 2001] and M. Atkin's work on the GRASP system that is applied to a capturc-thc-flag game [Atkin, 1998].

Computer soccer pursues similar goals and has become quite popular [Stone, 2002]. This domain can be regarded as a simplified RTS game with only a few objects, no economy, unremarkable terrain features, and more or less complete information. Another big difference is that in computer soccer a small number of agents are required to compute their actions autonomously whereas in RTS games a large number of objects have to be orchestrated.

The other body of literature relevant to this project is on military analyses and applications. Research in this area spans from mathematical combat models [Ilachinski, 1996] over computer generated forces - which are used in simulation and training - to decision-support systems that aid commanders and troops on the battle-field or even control entire weapon systems autonomously. Our project tries to bring both research communities together.

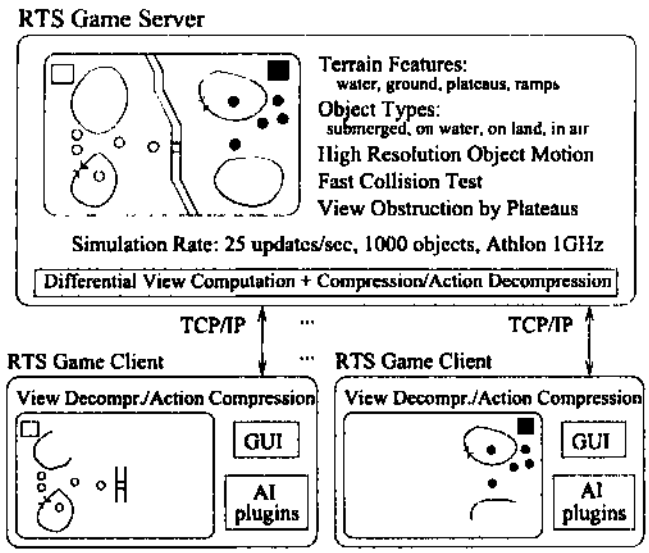

Figure 1: Server-client RTS game architecture. Clients connect to a central server which sends player views, receives actions for all objects, and updates the state of the world.

References

[Atkin, 1998] M.S. Atkin. AFS and HAC: Domain-general agent simulation and control. In *AAAI Workshop on Software Tools for Developing Agents,* 1998.

[Buro, 2002] M. Buro. ORTS: A hack-free RTS game environment. In *Proceedings of the Third International Conference on Computers and Games,* pages 156 161, 2002.

[Forbus *et al.,* 2002] K.D. Forbus, J.V. Mahoney, and K. Dill. How qualitative spatial reasoning can improve strategy game AIs. *IEEE Intelligent Systems,* 17(4):25 30, July 2002.

[Herz and Macedonia, 2002] J.C. Hcrz and M.R. Macedonia. Computer games and the military: Two views. *Defense Horizons. Center for Technology and National Security Policy, National Defense University,* 11, April 2002.

[Ilachinski, 1996] A. Ilachinski. Land warfare and complexity CRM 96-68, Center for Naval Analysis Research, 1996.

[Laird, 2001] J. Laird. Using a computer game to develop advanced AI. *Computer,* 34(7):70-75, July 2001.

[Stone, 2002] P. Stone. Multiagent competitions and research: Lessons from RoboCup and TAC. *RoboCup 2002 International Symposium,* June 2002.

[von der Lippe et al., 1999] S. von der Lippe et al. A robotic army: The future is CGF. In *10th Conference on Computer Generated Forces and Behavioral Representation,* Florida, USA, 1999.

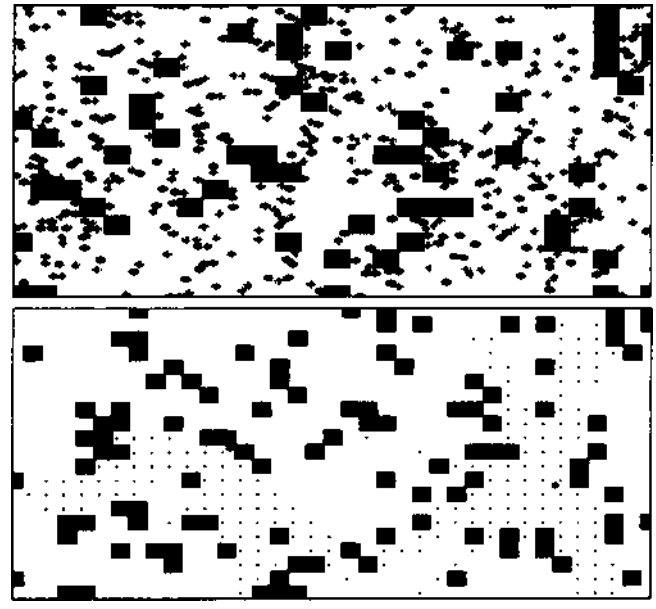

Figure 2: a) Motion/collision stress test: hundreds of moving ground and air units on hilly terrain. All units are visible to all players, b) Tile-based vision: views are obstructed by hills.