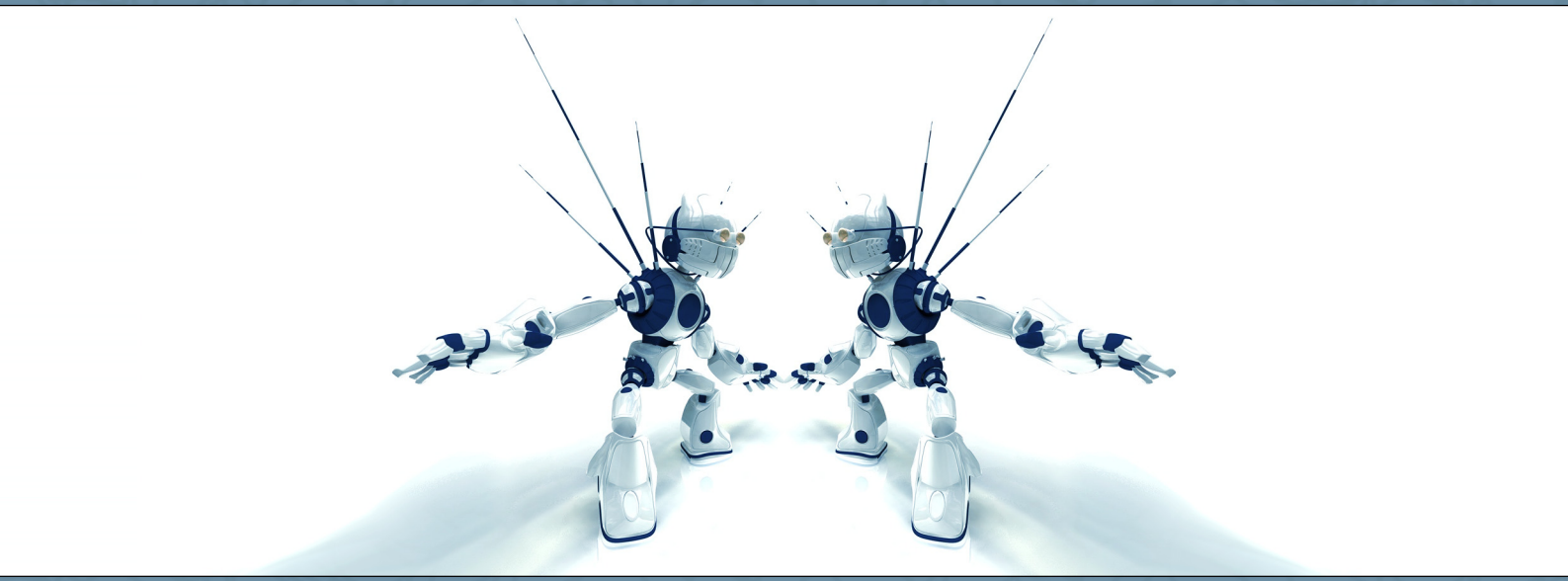
Universiteit Maastricht

Learning to Play as a Team

Designing an Adaptive Mechanism for Team-Oriented Artificial Intelligence



Sander Bakkes

ikat
institute for knowledge and agent technology

Learning to Play as a Team

Designing an Adaptive Mechanism for
Team-Oriented Artificial Intelligence

Sander Bakkes

Master's Thesis CS 03-04

Thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science of Knowledge Engineering
in the Faculty of General Sciences
of the Universiteit Maastricht

Thesis committee:
Prof. dr. H.J. van den Herik
Prof. dr. E.O. Postma
Ir. P.H.M. Spronck
Drs. H.H.L.M. Donkers

Universiteit Maastricht
Institute for Knowledge and Agent Technology
Department of Computer Science
Maastricht, The Netherlands
November 2003

"If Pacman had affected us as kids we'd be running around in dark rooms, munching pills and listening to repetitive music."

Marcus Brigstocke, British comedian

Contents

Preface

There is a theory which states that the most powerful problem-solver in the universe is the human brain, that created "the wheel, New York, wars and so on" (after Douglas Adams [1]). There is another theory which states that the most powerful-problem solver in the universe is the evolutionary mechanism, that created the human brain (after Charles Darwin [11]).

For centuries on end, mankind marvelled its self-proclaimed intelligence, but at present we have not even managed to create an artificial entity with the intellectual capabilities of an ant. For instance, ever since the creation of the first team-oriented commercial computer games, the artificial intelligence of game opponents lacked adaptive capabilities, e.g., the ability to learn from mistakes. A mechanism suitable for adaptive behaviour in team-oriented commercial computer games does not exist.

This thesis discusses the design of the *Tactics Evolutionary Adaptability Mechanism* (TEAM), an evolutionary inspired mechanism for adaptive team-oriented artificial intelligence. We expect researching the topic of adaptation in artificial environments to be beneficial to the study and application of artificial intelligence techniques in general, and machine-learning techniques in commercial computer games in particular.

I would like to thank the Institute for Knowledge and Agent Technology (IKAT) in Maastricht for allowing me to pursue my interest in the field of artificial intelligence and granting me the freedom to fill in the master's thesis research as I deemed fit. My special gratitude goes out to my coach, Pieter Spronck, for his support, dedication and being a constant source of inspiration.

Outside the academia, living in Maastricht offered many beautiful and eventful days, and I consider myself lucky for sharing them with dear friends.

Finally, I would like to express my immeasurable appreciation to my parents for their unconditional devotion, support and faith.

Sander Bakkes
Maastricht, November 2003

Chapter 1

Introduction

Freely adapted from Arthur C. Clarke [9], one could claim that through the advancement of artificial intelligence (AI) any sufficiently advanced creation will be indistinguishable from a human being. However, observing current robots being unable to find their way in a sparsely furnished room, pats convincing AI in the far future, if it is achievable at all. According to Darwinistic beliefs, mankind itself once was as undeveloped as these artificial creations, and yet, humanity advanced. According to Tim Robbins [36], humanity advanced 'not because it has been sober, responsible, and cautious, but because it has been playful, rebellious, and immature'.

With this element of playfulness in mind, this chapter presents the background of the thesis. In section 1.1 the subject of the thesis is introduced in a historical perspective. Section 1.2 discusses the background of the thesis' research and section 1.3 discusses the research objectives. An outline of the thesis is given in section 1.4.

1.1 The evolution of commercial computer-game AI

The designers of 'ancient' commercial computer games already acknowledged the need for computer-controlled opponents to show pseudo-intelligent behaviour. From an entertainment point of view there was no need for this behaviour to be comparable to human intelligence. Yet, one way or the other the computer-controlled opponents should be intelligent enough to entertain the person that is playing the game.

A classic example of entertaining AI in commercial computer games, is the game Pacman (illustrated in figure 1.1). This game implements a rudimentary form of AI that ensures entertaining gameplay. The opponents of the game Pacman moved randomly through the environment with an increasing speed. However, there are many facets of the AI in Pacman that can be improved. In an improved approach the opponents would, for instance, conspire against the

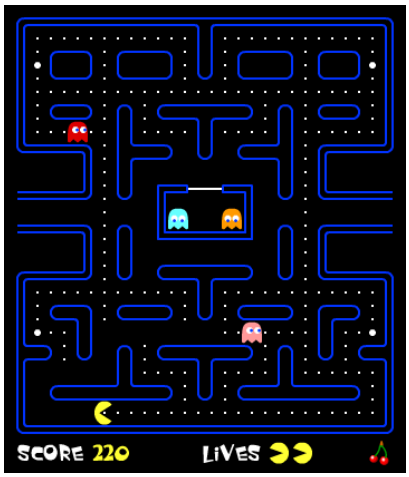


Figure 1.1: Screenshot of the game Pacman.

human player in a massive organised manhunt. Such an improvement would certainly be challenging, perhaps even too difficult. Therefore, the designers of commercial computer games seek a balance between on the one hand improved AI and on the other hand entertainment.

Nowadays, the digital entertainment industry is continuously striving for realistic and stunning audio-visual presentation. Unfortunately, the incorporation of artificial intelligence does not yet get the attention it deserves. Artificial intelligence is one of the elements of so-called 'gameplay', which comprises everything but the visual and auditory presentation of the game. A game needs interesting gameplay to keep the player interested when after the first few minutes of play the initial amazement fades away. Therefore, the element of gameplay should receive considerable attention from the developers. However, even in high-profile games, such as the recently released first person shooter Epic's UNREAL II, designers often focus on the presentation of the game, and refrain from designing an exceptional gameplay. In recent years, game reviewers have critically examined the artificial intelligence of commercial computer games in addition to sound, graphics and gameplay issues. More and more they emphasise the need of improved artificial intelligence in commercial computer games.

To fulfil the need of improved artificial intelligence, our research discusses a step towards adaptation of opponent behaviour capable of exceeding the limitations of its designer's vision by, for instance, generating new unforeseen tactics and intelligently adapting to the human player in order to create a more challenging experience.

1.2 Research background

The research background of this thesis is artificial intelligence in commercial computer games. Regarding the research background, we discuss two observations: the inferior artificial intelligence in current commercial computer games in general, and the lack of adaptive team AI in commercial computer games in particular.

First, in recent years the opponents in commercial computer games respond to their environment in an increasingly intelligent way. Commonly, the artificial intelligence of commercial computer-game opponents is based on non-adaptive techniques [44]. Non-adaptive artificial intelligence has a major disadvantage: once a weakness in the artificial intelligence is discovered, nothing stops the human player from taking advantage of the discovery. The disadvantage can be resolved by letting commercial computer-game opponents behave adaptively, e.g., learn from their mistakes. Adaptive behaviour can be achieved by using machine-learning techniques. Examples of machine-learning techniques are artificial neural networks [27] and evolutionary algorithms [16]. In future-generation games, machine-learning techniques can be used to improve the artificial intelligence of computer-game opponents.

Second, the organisation and interaction of computer-game opponents in team-oriented games is a challenge for artificial-intelligence research. This so-called 'team AI' is not entertaining in the current generation of games, with regard to the preference of human players to play against other humans over playing against artificial opponents [35]. A human team is easily able to achieve excellent results against a team of computer controlled opponents. Currently, the team AI in commercial computer games lacks adaptive behaviour. We focus the research on investigating how to improve team AI by using machine-learning techniques. The evolutionary algorithms machine-learning technique is used as main inspiration, since we expect it to be suitable for our purpose. This expectation is, for a large part, based on the impressive results that evolutionary algorithms have shown in the field of robot soccer [29] and coevolutionary games [12]. Nowadays, the behaviour of opponents in commercial team-oriented computer-games is at best a superficial attempt to convince an observer of their intelligence. In practice the artificial intelligence of commercial computer-game opponents is not as developed as one would wish. In future games we envision artificial intelligence that is capable of adaptive behaviour that resembles intellectual capabilities of human intelligence. To ultimately teach artificial creations how they can learn is strived for. This thesis presents one step into this direction.

1.3 Problem Statement and Research Question

As stated above, the background of our research is artificial intelligence in commercial computer games. We observed that current commercial computer-game opponents are endowed with inferior team-oriented behaviour. The observation

has led us to the following problem statement.

> **Problem statement:** *Is it possible to improve the performance of opponents in state-of-the-art commercial computer games, with regard to their team-oriented behaviour?*

As stated in section 1.2, one reason for the inferiority of team-oriented behaviour in commercial computer games is that it lacks adaptive team AI.

We aim at creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment.

An approach to deal with this aim is to create a mechanism which imposes adaptive team AI on commercial computer-game opponents. The following research question guides our research.

> **Research question:** *Is it possible to create a mechanism that imposes adaptive team AI on commercial computer-game opponents and achieves a qualitatively acceptable performance?*

In our attempt to answer the research question, we have three objectives.

1. Designing a mechanism that imposes adaptive team AI on opponents in commercial computer games.
2. Implementing the design in a test environment.
3. Obtaining a qualitatively acceptable performance of the adaptive mechanism, i.e., performance that is computationally fast, robust, efficient and effective [38].

1.4 Thesis overview

The remainder of the thesis is organised as follows. Chapter 2 discusses an overview of recent developments on artificial intelligence in commercial computer games. Chapter 3 discusses the first research objective; designing a mechanism that imposes adaptive team AI on opponents in commercial computer games. Subsequently, chapter 3 discusses the second research objective; implementing the design in a test environment. Chapter 4 discusses performing experiments, preparatory to discussing the third research objective; obtaining a qualitatively acceptable performance of the adaptive mechanism. Chapter 5 discusses the third research objective, and additionally discusses the approach. Chapter 6 first answers the research question and then comes to a conclusive answer of the problem statement.

Chapter 2

AI in commercial computer games

This chapter discusses recent developments on artificial intelligence in commercial computer games. The chapter is structured as follows: section 2.1 discusses (the lack of) adaptive behaviour in commercial computer games. In section 2.2, the topic of AI in commercial computer games is extended, by discussing team AI in detail. Subsequently, section 2.3 discusses team AI in a typical commercial computer-game. At the end of this chapter a summary is presented.

2.1 Adaptive behaviour in commercial computer games

Commercial computer-game opponents who learn, or exhibit adaptive behaviour, are rarely investigated. As yet adaptive behaviour is not applied in practice because it cannot be accomplished using conventional methods. Adaptive behaviour leans on techniques that are still unproven in commercial computer games. The application of unproven adaptive techniques is risky for game developers because of two reasons: because it can slow down their development cycle, and because the end-results are typically unpredictable [43]. Game developers therefore tend to reuse proven, non-adaptive, techniques. We briefly discuss two commonly used proven techniques: scripts and finite-state machines.

First, scripts are typically used to implement the artificial intelligence of commercial computer-game opponents. Unfortunately, the scripting technique lacks adaptive capabilities. Therefore, it is not possible to design an exhaustive script. Once a human player discovers a 'hole' in the script nothing prevents the player from exploiting it. The discovery made public, can spoil the game for the gaming community.

Second, finite-state machines are commonly used to model intelligence. An example of a finite-state machine is displayed in figure 2.1 [45]. Finite-state ma-

chines (commonly abbreviated as FSMs) can be described by four components: (1) an initial state or record of something stored someplace, (2) a set of possible input events, (3) a set of new states that can result from the input, and (4) a set of possible actions or output events that result from a new state. Finite-state machines entail the disadvantage to offer no possibility to adapt its behaviour. New behaviour requires the finite-state machine to be redesigned.

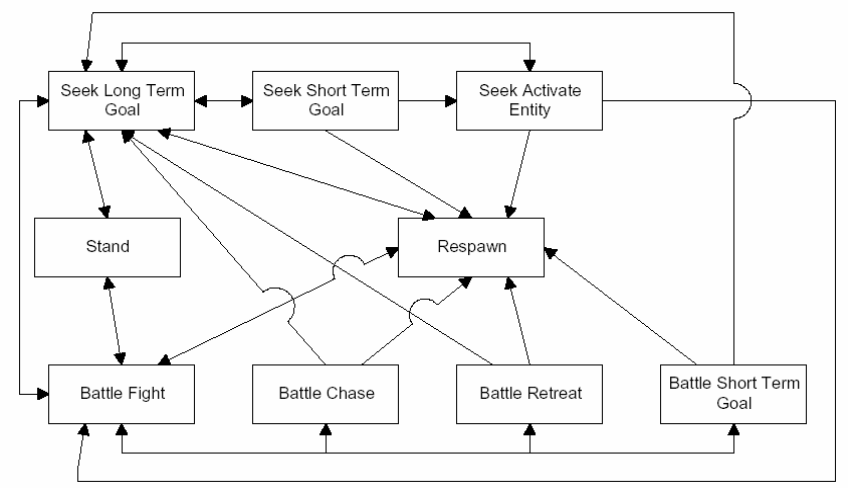


Figure 2.1: Finite-state machine of a Quake III agent.

Alternatives for non-adaptive techniques exist, and are demanded by a need in the market to employ adaptive AI in commercial computer games [42]. To prevent exploitation of inferior behaviour of computer-game opponents, adaptive behaviour is desirable. Online adaptability, which can be used to learn from mistakes, is a key feature many game developers yearn for. Some existing commercial computer-game opponents appear to adjust their behaviour to the human players' actions, but often programmers of a game rely on controlled randomness [20]. This form of adaptation is limited. Genuine adaptive intelligence lies, for instance, in the ability to learn from mistakes and to find new ways of accomplishing a goal. Machine-learning techniques, such as classifier systems, neural networks and evolutionary algorithms, are specifically designed for adaptive AI. Therefore, machine-learning techniques are potentially suitable for application in commercial computer games.

2.2 Team AI in commercial computer games

Team-oriented AI is a field of research, originating from the need to let agents [18], in our case commercial computer-game opponents, compete or cooperate in a specific environment. In the remainder of this thesis we study the cooper-

ation of agents to accomplish specific tasks. Adaptive team AI in commercial computer games consists of four components:

1. **Individual agent AI.** This components is required because each agent needs appropriate rudimentary intelligence for a specific environment. A practical example: agents would be useless should they be running into traps like lemmings. Individual agent AI is game-specific. Therefore, we do not discuss it in more detail.

2. **Means of communication.** This component is discussed in sub-section 2.2.1.

3. **Team organisation.** This component is discussed in sub-section 2.2.2.

4. **An adaptive mechanism.** This component allows for adaptive behaviour. Note that without an adaptive mechanism, team AI is non-adaptive. Because of the importance an adaptive mechanism, we devoted a whole chapter to it (chapter 3).

2.2.1 Means of communication

Analogous to human players, who need to communicate with each other in order to establish social interaction, communication is also a requirement for agents in team-oriented environments. They need to come to a mutual understanding. Typically, agents pass along messages containing information or orders. For example, an agent informs a team-mate of the occurrence of a pre-defined event. Likewise, information can be used to compute needed counteractions and spread orders amongst the agents of a team.

Communication in team AI is not limited to creating and propagating messages. It is imperative that the semantics of a message are captured, so agents can process the messages in the way the sender of the original message expects. For instance, often a technique called 'match templates' is used, which allows agents to 'understand' a received message. Once the meaning of a message is clear, the agent gives an appropriate response, e.g., an action inferred from its rule-base.

In case only agents have to communicate with each other, message interpretation is a redundant task since it is algorithmically defined in the source-code of the game. However, human players often control agents by sending messages. It is conceivable that an agent receives an indistinctive message from a human player, and is expected to respond in an intelligent way. In case of misunderstood chat messages Eliza-like responses are a possibility [47]. It is quite common that a game requires complex message interchange. Therefore, team-oriented games benefit from the implementation of a well-defined communication and transaction protocol, e.g., from the FIPA protocol suite [18].

2.2.2 Team organisation

Team AI requires the design of a team organisation, since there can be no team cohesion without it. There are two distinctive approaches to organise a team of agents in commercial computer games: namely (1) a decentralised approach, and (2) a centralised approach.

The decentralised approach is a small extension to the already existing individual agent's AI. This approach is usually modelled as displayed in figure 2.2 (left). Freely adapted from the adage that 'the intelligence is in the environment, not in the ant' [35], the decentralised approach is based on the theory that team behaviour emerges when individual agents exchange observations and intentions. For team behaviour to emerge, the agents need to base their decisions not merely on their own internal state but also on the information received from other agents. Since this requires the interchange of information, basic means of communication are needed, as mentioned in sub-section 2.2.1. Next to the relative ease of implementation, a decentralised organisation has shown impressive results [40]. A disadvantage of the decentralised approach is that it needs conflict resolution. Messages that are propagating across the agent-network represent different intentions and suggestions. All these intentions and suggestions typically generate conflicts. Another disadvantage of the decentralised approach is that there is no process 'thinking' for the team as a whole. Therefore the actions of each agent are fully autonomous. Fully autonomous agents are useful in specific environments, but generally the lack of team cohesian is considered as a disadvantage since it does not allow for tight coordination of agents.

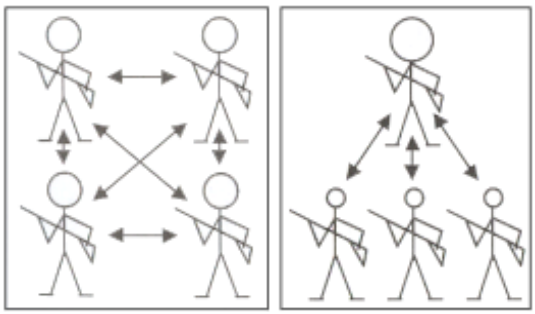


Figure 2.2: Decentralised organisation (left) and centralised organisation (right).

The centralised approach does not cope with the disadvantages of the decentralised approach. This approach is schematically displayed in figure 2.2 (right). The centralised approach is specifically designed to create and maintain well-organised team cohesion. In this approach, decision making is centralised. A decision is processed into orders, which are sent to agents. Borrowed from the military command hierarchy, the centralised approach enables fast agent responses and coordinated manoeuvres without wasting time exploring possibilities and resolving conflicts. Within a centralized approach we can distinguish

two basic implementations: namely an authoritarian command style, and coaching command style.

The authoritarian command style is only focussed on team results and ignores extra agent information. The waste of information can result in bad decisions for some agents. However, the authoritarian command style is highly efficient for issuing commands to agents which are treated as ignorant soldiers. The coaching command style is a less strict way of commanding agents. It is aimed at advising an agent what to do, rather than forcing him to orders. The individual agents can choose to give no priority to an advice. A coaching command style is only suitable if a decreased team cohesion is of no significance. Additionally, one can define command styles which borrows from both the authoritarian command style and the coaching command style, where some commands are interpreted as orders and others as suggestions.

2.3 Adaptive team AI in a typical commercial computer game

This section discusses adaptive team AI in a typical commercial computer game, in advance of designing a team-oriented adaptive mechanism for commercial computer games.

A typical team-oriented game is the Capture-The-Flag (CTF) team-based game mode of the commercial computer-game Quake III, a real-time action game. After a brief description of the game, we discuss the Quake III implementation of the four components required for adaptive team AI: (1) individual agent AI, (2) means of communication, (3) team organisation, and (4) an adaptive mechanism.

The Capture-The-Flag team-based game mode is found in practically all first-person shooters. Van Waveren gives an adequate description of the game and the Capture-The-Flag team-based game mode [46]:

> *"Quake III Arena belongs to the genre of the first person shoot-em up games. A player views from a first person perspective and moves around in a real-time 3D virtual world. The most important tasks are staying alive and eliminating opponents within this virtual world. These opponents are other people, equal in strength and abilities, connected to the same game through a network or the Internet. The players have a wide range of weapons, items and power-ups available to aid in the battles. The game has a set of different virtual environments called levels or maps, that contain rooms and hallways. The battles in the game take place in these maps much like gladiators fight in an arena. Players can score points by taking out other players. When killed, a player respawns at one of the designated locations on the map and can continue to fight. Quake III Arena also has several team-oriented gameplay modes. In normal teamplay there are two teams with players that fight each other. The team with the highest*

accumulated score, of all players on that team, wins. There is also a Capture-The-Flag (CTF) team-based game mode. Again there are two teams with players. Each team has a base structure in the game world or map. A flag is positioned in such a base. A team scores points by capturing the flag of the opposing team and bringing it back to their own flag in their own base."

First, individual agent AI is provided to each Quake III agent, allowing it to observe the environment it finds oneself in, navigate through it by using A* based pathfinding techniques [13][28], and providing means of survival in the form of combat and tactical skills. Agents are assigned to a profile of behavioural characteristics and a difficulty level. These two assignments define behavioural elements, such as, the accuracy the agents shoot with. Quake III agents are based on a layered architecture [3], which is displayed in figure 2.3 [46]. The first layer is the primary input and output layer for the agent. The second layer provides rudimentary intelligence, such as a manoeuvring capability. The third layer defines 'intelligent' behaviour, such a shooting a weapon. The fourth, and to our research most significant, layer defines the *team control mechanism*, which issues team-oriented agent commands.

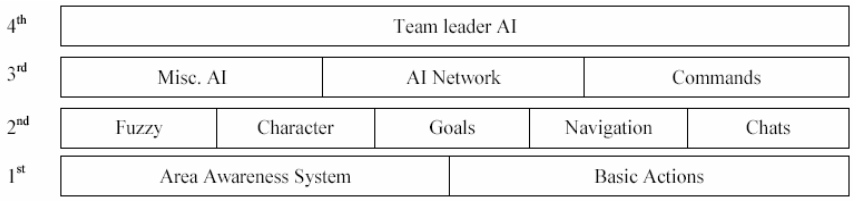


Figure 2.3: Quake III agent layered architecture.

Second, means of communication is made possible by the ability of agents to send messages. The agents use match templates to capture the semantics of a message. When agents are communicating with human players, Eliza-like responses are used [47].

Third, regarding team organisation, the designers of Quake III chose a centralized approach. The centralised team organisation approach is implemented in the form of a coaching command style. Therefore, as explained in section 2.2.2, the team organisation in Quake III does not require conflict resolution.

Fourth, an adaptive mechanism is not present in Quake III. The team AI uses a static rule-base and a simple finite state machine which represents the possible states of the team game. The team AI assigns each of the agents an offensive, a defensive, or a roaming role. The offensive and defensive roles are global tactical assignments. The roaming role does not entail a global tactical purpose and results in reactive agent behaviour. The implementation of a role is different in each state. For instance, in state x a defensive role means that an agent receives the command to defend the friendly flag, while in state y

Team-members	Defensive agents	Offensive agents
1	0	1
2	1	1
3	1	2
4	2	2
...	...	...
$n-1$	$\frac{n}{2}-0.5$	$\frac{n}{2}+0.5$
n	$\frac{n}{2}$	$\frac{n}{2}$

Table 2.1: Example of static division of roles for one state.

this defensive role implies that an agent is commanded to accompany a team-member. In Quake III, the division of roles for each state is static with respect to the number of team-members. An example of this static division of roles for a specific state is displayed in table 3.1, where n is an even positive integer.

The CTF team-based game mode requires strategic interplay of agents. Since Quake III lacks an adaptive mechanism, required for adaptive team AI, we use the game Quake III, and its CTF team-based game mode, as a basis for designing and testing an abstract and generally applicable team-oriented adaptive mechanism.

Figure 2.4: Agent capturing a flag in the Quake III CTF game.

2.4 Summary

In this chapter three topics were discussed, which concerned artificial intelligence in commercial computer games.

First we discussed the lack of adaptive artificial intelligence in commercial computer games. We noted that machine-learning techniques are designed for adaptive intelligence, and therefore have the potential to be successfully applied in commercial computer games.

Second, we discussed team AI in commercial computer games, and stated that adaptive team AI consists of four components:

1. individual agent AI,
2. means of communication,
3. team organisation, and
4. an adaptive mechanism.

Third, we discussed adaptive team AI in Quake III, a typical commercial computer game. We observed that the game lacks an adaptive mechanism, required for adaptive team AI. Therefore, we decided to use this game as a basis for designing and testing an adaptive mechanism that imposes adaptive team AI on opponents in commercial computer games. This adaptive mechanism is discussed in chapter 3.

Chapter 3

The TEAM artificial intelligence adaptation mechanism

This chapter discusses the first research objective; designing a mechanism that imposes adaptive team AI on opponents in commercial computer games. Subsequently, the chapter discusses the second research objective; implementing the design in a test environment.

We designed the Tactics Evolutionary Adaptability Mechanism (TEAM), an adaptation mechanism that imposes adaptive team AI on opponents in commercial computer games. TEAM is explicitly designed to be a generic adaptive mechanism for team-oriented commercial computer games in which the game state can be represented in a finite state machine. The design is illustrated by projecting design choices to the game Quake III, a typical commercial computer-game.

As discussed in chapter 1, the adaptive mechanism is based on evolutionary algorithms. The first section (3.1) of this chapter presents a description of evolutionary algorithms. Subsequently, section 3.2 presents a synopsis of TEAM. After the synopsis we discuss three design elements of TEAM: the team-oriented adaptation approach (section 3.3), the design of the team-oriented evolutionary algorithm (section 3.4), and fail-safe considerations of TEAM (section 3.5). The implementation of TEAM in Quake III is briefly discussed in section 3.6. A summary of the chapter is presented in section 3.7.

3.1 Evolutionary algorithms

In the preface of this thesis we already mentioned that there is a theory which states that the most powerful problem-solver in the universe is the human brain, that created "the wheel, New York, wars and so on" (after Douglas Adams [1]).

There is another theory which states that the most powerful problem-solver in the universe is the evolutionary mechanism, that created the human brain (after Charles Darwin [11]).

Conceptually, evolutionary algorithms are inspired by the Darwinistic evolution theory on the origin of man and on the Mendelian genetics. We use evolutionary algorithms as a basis for designing a machine-learning technique for team-oriented artificial intelligence. Eiben introduces evolutionary algorithms as follows [16]:

> *"Given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population. It is easy to see such a process as optimisation. Given an objective function to be maximised we can randomly create a set of candidate solutions, i.e., elements of the objective function's domain, and apply the objective function as an abstract fitness measure - the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is a binary operator applied to two selected candidates (the so-called parents) and results one or two new candidates (the children). Mutation is unary, it is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete - based on their fitness - with the old ones for a place in the next generation. This process can be iterated until a solution is found or a previously set computational limit is reached. In this process selection acts as a force pushing quality, while variation operators (recombination and mutation) create the necessary diversity. Their combined application leads to improving fitness values in consecutive populations, that is, the evolution is optimizing."*

A schematic representation of the evolutionary process is displayed in figure 3.1 [17]. Eiben notes that evolution actually is not 'optimizing', it is 'approximating', by approaching optimal values closer and closer over its course.

3.2 Synopsis of TEAM

This section presents a synopsis of TEAM, in advance of discussing it in detail. TEAM is based on evolutionary algorithms. However, TEAM employs four features which distinguish the adaptive mechanism from typical evolutionary approaches:

1. **Centralised agent control mechanism evolution.** TEAM evolves an agent control mechanism, contrary to evolving a population of individual agents. In section 3.3 this first feature is discussed in detail.

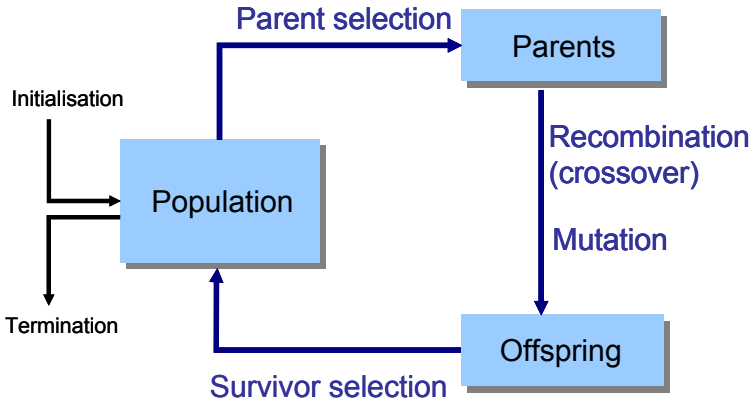


Figure 3.1: The evolutionary mechanism cycle.

2. **Mutualistic-symbiotic evolution.** Team-oriented behaviour is learned by a cooperation of multiple instances of an evolutionary algorithm. Each instance of the evolutionary algorithm learns relatively uncomplicated team-oriented behaviour. The combination of all instances, results in relatively complex team-oriented behaviour. In section 3.4 this second feature is discussed in detail.

3. **State-specific genome evolution.** Each instance of the evolutionary algorithm evolves a population of relatively short genomes, which represent state-specific team-oriented behaviour, contrary to evolving a population of relatively large genomes, which represent behaviour for every state. This feature implied that the design of a perspicuous, generally applicable, evaluation function is feasible. In section 3.4 this third feature is discussed in detail.

4. **Evolution with history fall-back.** This feature ensures a desirable evolutionary course. In section 3.5 this fourth feature is discussed in detail.

Exceptionally, TEAM is the first existing team-oriented artificial intelligence adaptation mechanism.

3.3 Approaches for team-oriented adaptation

In academic publications and existing commercial computer games we have not encountered an adaptive mechanism for team AI. Therefore we have to design an adaptive mechanism ourselves. Based on evolutionary algorithms, there are two approaches for creating a suitable adaptive mechanism for adaptive team AI, similar to the (de)centralised team organisation approach discussed in section 2.2.2. First, the decentralised approach, and second, the centralised approach.

The decentralised approach is based on standard evolutionary algorithms concerning the evolution of individual agents. A decentralised adaptation approach is aimed at evolving the AI of a population of individual agents capable of operating in a team and by evolutionary influences performing this task with increasing efficiency.

The centralised approach has a viewpoint directed at evolving the AI of the team itself, rather than the AI of individual agents. The centralised adaptation approach is derived from the desire of directly to evolve to improved behaviour for the team as a whole. Thus, the centralised approach typically does not evolve each team-member, but evolves whatever controls this team.

One could state that team-behaviour already is emerging from the internal organisation and coordination of agents [50]. Thus, instead of evolving agents, evolving an abstract mechanism that controls these agents is a straightforward derivation from this observation, resembling the centralised approach. A disadvantage of the decentralised approach is the expected complexity of evaluating the team's behaviour on the level of each individual agent. Ascribing a successful action to the effectiveness of one agent or the ineffectiveness of its opponent requires adding a significant amount of game specific information. The centralised approach offers a less complex and generally applicable evaluation since it only needs to evaluate how the team is behaving as a whole. As is discussed in sub-section 3.4.4, this is something that can be expressed with relative ease. Therefore, our preference for a suitable adaptive mechanism approach goes out to the centralised approach.

Our preference for a centralised approach is strengthened by an additional consideration. The decentralised approach is an approach for evolving a population of agents. Since this approach bears close resemblance with standard evolutionary algorithms we naturally have to consider their downsides, of which a computationally high cost and slow evolution are harmful to our research objective. In order for human players competing with the team of agents to notice behavioural adaptation, the evolution has to occur in a limited timeframe. The team behaviour in a decentralised approach has to emerge from the interplay of every individual agent, while a centralised approach is evolving the behaviour for the whole team directly. We therefore expect an adaptive mechanism that is based on a centralised approach to evolve faster than an adaptive mechanism that is based on a decentralised approach.

3.4 Design of the team-oriented evolutionary algorithm

In this section we present the design elements of TEAM's evolutionary algorithm. Derived from common steps for building an evolutionary algorithm we denote six elements [16]:

1. representation,
2. population,

3. parent selection mechanism,

4. evaluation function,

5. variation operators,

6. survivor selection.

In the following sub-sections we present a detailed description of our design choices for each of the above.

3.4.1 Representation

We observe that the team control mechanism for the existing Quake III implementation is based on a FSM with four states (figure 3.2 [45]). These four states are team-game specific; in this case they represent the states in a CTF game. The state transitions that are displayed are all possible transitions for this CTF game. Each state describes a hard-coded tactic merely for the specific state. For example in state x a programmer can incorporate tactic x, and in state y a different tactic (y). As discussed in section 2.3, in the Quake III implementation the description of a tactics consists of a division of roles, which can be represented by real valued genes.

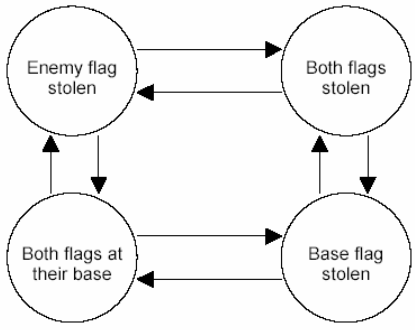


Figure 3.2: Four states of a CTF game.

A common approach to represent tactics for use in an evolutionary algorithm would be to create one long genome, which contains the required genetic information for all states. We, however, choose to deviate from this approach and explore something different. We propose two deviations from standard evolutionary algorithms, which we discuss in the upcoming paragraphs.

Our first design choice is to base our genomes on the tactic description of merely one state. Thus, for every state of the FSM we generate different genomes. The genomes for each state have no communal relation with one another.

Our second design choice is to have multiple instances of the evolutionary algorithm, one for every state of the FSM.

A primal consideration for the first design choice, basing the genomes on the tactic description of one state, is that it allows us to keep the genomes relatively small. To represent a tactic description of a state, a genome requires a few real valued genes which bear the division of roles description. By using small genomes we are able to evolve quickly. The genome no longer represents the inter-state cohesion, which is favourable in our case, since the actions that occur in each state can be completely different and have no direct relation with one-another.

Regarding the second design choice, to have multiple instances of the evolutionary algorithm, a benefit from this design choice is that it allows short evolutionary cycles. Each state transition completes one cycle of a state's evolutionary algorithm. While the real-time environment is in a specific state, the evolutionary algorithms of other (idle) states are accessible for computations. One could, for instance, already mutate the population of idle state's genomes. All instances of the evolutionary algorithm are operating independently. Each instance of the evolutionary algorithm is focussed at learning one relatively small task, and is able to learn this task without intervening with the learning tasks of other instances of the same evolutionary algorithm.

As we mentioned in the previous paragraph, each state transition completes one cycle of a state's evolutionary algorithm. The game's state transitions occur very often, an average of one transition in five seconds is quite common. It is our expectation that frequent state transitions in combination with our design choices result in a human player to quickly notice the effect of the evolutionary process.

3.4.2 Population

In our environment, designing the population implies we have to define how many genomes are in the population. Since the game is running in real-time we can only evaluate one genome at a time. Therefore, the speed of the evolution benefits from a small population size. In our design we use a population size of 5.

In typical evolutionary algorithms the first population is a seed of randomly generated genomes. In our environment however we do not have the luxury of allowing randomly generated genomes to evolve to desired behaviour since someone who is playing the game should not have to wait for the opponent team to become challenging in time, it has to be challenging right from the start of the game. To ensure that the strength of a team is at least equal to what has been manually designed, our population is initialised with genomes describing a strategy close to each state's manually designed strategy. By doing so we enforce that a human team is not confronted with an inferior opponent-team when the game starts.

Consequently, care should be taken that the evolutionary adaptive mechanism offers the capability to drastically deviate from this initial strategy in order

to provide a dynamic and unrestricted evolution. Driving force to accomplish this capability lies in a suitable parent selection mechanism (sub-section 3.4.3), choosing appropriate variation operators (sub-section 3.4.5), and designing an adequate survivor selection mechanism (sub-section 3.4.6).

3.4.3 Parent selection mechanism

Our population consists of a small number of genomes. It therefore is logical that of the given population we design our parent selection mechanism to use the best genome to control the team behaviour. By selecting the best genome as parent, the next population contains genomes based on this parent, which is expected to result in equal or better team-behaviour.

Naturally, a small population size is dangerous with respect to genetic diversity. For lack of genetic diversity we need means of escaping local optima, which is discussed in sub-sections 3.4.4, 3.4.6 and 3.5.2.

3.4.4 Evaluation function

An essential element of an evolutionary algorithm is a function which evaluates the quality of a genome, which is expressed by the so-called fitness value. In evolutionary algorithms such a function is denoted as an evaluation or fitness function, depending on the terminology of the research environment. Implicitly, the evaluation function represents the requirements the genomes have to adapt to. In our environment, evaluating genomes is complex because of the environment's real-time nature and since the assessment of agent's behaviour in the CTF team-based game mode is not transparent.

The design of our evaluation function uses an abstract and generally applicable view of the game-state. Our evaluation function consists of three components: a base fitness value, a time-scaling mechanism, and a delayed-reward mechanism [27].

First, the base fitness value is calculated by using annotations on the finite state machine, which describes the states of a game (displayed in figure 3.2). Generally speaking, if the application use a genome that provides an undesirable state transition, this genome should receive a low fitness value. Similarly, a desirable state transition should be rewarded with a high fitness value. By using such an orderly and straightforward basis for the evaluation function, we are not faced with complex questions concerning the best behaviour in a certain state. Analogously, we annotate the FSM of the team-oriented game, as is displayed in figure 3.3, where a desirable state transition (denoted with $+$) receives a *base* fitness value of 1.0, and an undesirable state transition (denoted with $-$) receives a *base* fitness value of 0.0. Note that, depending on specific conditions, certain state transitions can be either desirable or undesirable. In the displayed annotated FSM, 'diagonal' transitions are greyed out. They are theoretically possible, but since we cannot determine via what route the transitions would normally have taken place, these transitions receive a neutral fitness value to make sure they do not affect other genomes.

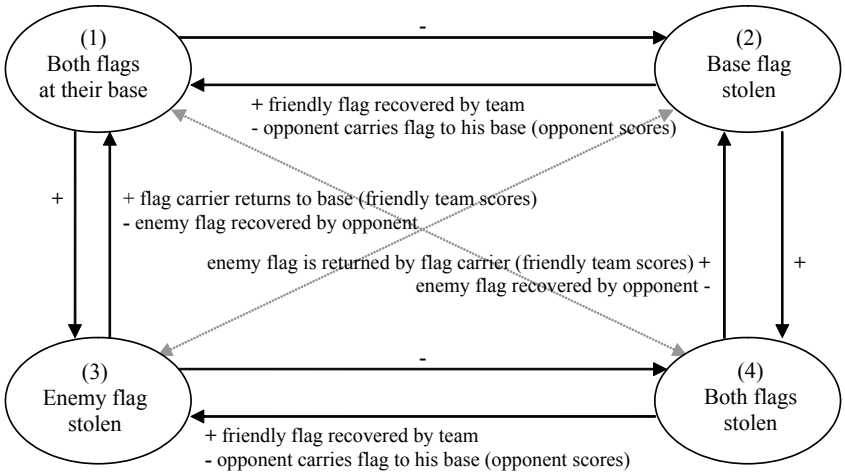


Figure 3.3: Annotated CTF finite state machine.

Transition	**Time to state transition**	**Fitness value**
+	Short	High
+	Long	Fairly high
−	Long	Fairly low
−	Short	Low

Table 3.1: Global characteristics of the evaluation function.

Second, a time-scaling mechanism is needed since, as one can intuitively expect, a black-and-white annotation of the FSM is not sufficient for measuring the success of a genome. If the game resides in a specific state for a long time one can consider this as neutral behaviour, since it neither transits to a desirable nor to an undesirable state. If we use this 'neutral' behaviour as a point of reference, we can use it to more accurately distinguish between desirable and undesirable behaviour. Note that in the FSM of the team-oriented game, a neutral state does not exist, which is typical for this type of game. However, it is possible to use the principle of neutrality by adding a time-aspect to the fitness function. Such a time-aspect is implemented with a time-scaling function, which scales the base fitness according to the amount of time for a state transition to take place. In our environment we want to demarcate between fast and slow occurrences of state transitions. The function should therefore have a significant effect on transitions that took less than a couple of seconds. It also should have a clear effect on transitions of various long time differences, e.g., in the interval of one to several minutes. We decided to use a damping square root function, of which the effect is plotted in figure 3.4.

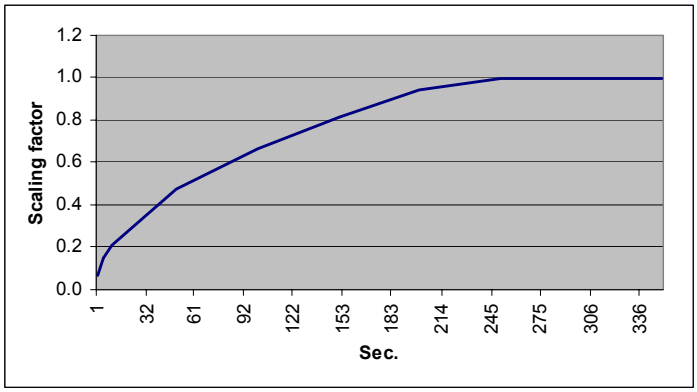


Figure 3.4: Output of the scaling function.

Third, a delayed-reward mechanism is required since without it the problem of creating local optima would arise. An illustrative example of local optima can be found in soccer, where fully defensive tactics are impractical for scoring goals, and fully offensive tactics are unsuitable for defending your own goal; tactics should be found that are effective in all situations. Consider that behaviour is only desirable if the team can retain this behaviour, or improve on this behaviour. If desirable behaviour quickly transits to undesirable behaviour, then the genome which resulted in the originally desirable behaviour does not suffice. Therefore, we need the evaluation function to encourage a desirable state transition at zero depth, but at the same time discourage an undesirable state transition at depth one. What is more, it is possible that a genome at zero depth is in itself good, but a genome at depth one is not adequate. The tactic deemed optimal for a specific state can be an unfortunate choice for a successive state. We therefore only reward genomes with a high fitness if their resulting desirable behaviour is not immediately annulled, because otherwise we are creating local optima. To avert the evaluation function from creating local optima, we use a delayed-reward mechanism to consider the long-term effect of the tactic described in a genome.

Concluding our design considerations, we propose the following evaluation function. Expression 3.1 denotes the fitness of a genome at a certain depth. Let $Fitness_i$ be the fitness of a genome at depth i. Then,

$$Fitness_i = \frac{1}{i+1}\left(base \pm \min\left(\frac{\sqrt{\text{sec}} - \sqrt{\frac{\text{sec}}{3}}}{10}, 1.0\right)\right) \tag{3.1}$$

where, i is the depth, *base* is the transition's desirability value as assigned by the annotated finite state machine, and *sec* denotes how many seconds were between the previous and the current state transition. The $\pm$ sign implies a $-$ operator for a desirable state transition, and a $+$ operator for an undesirable

state transition. Next, we denote the evaluation function in expression 3.2. Let $Delayed\_fitness$ be the delayed-fitness value of a genome. Then,

$$Delayed\_fitness = \sum_{i=0}^{n} Fitness_i \tag{3.2}$$

where i is a counter for the depth, n is a positive integer, and $Fitness_i$ the fitness of a genome at depth i. In our case $n = 2$, since we use a two-deep delayed-reward mechanism.

3.4.5 Variation operators

For online adaptation of team AI to be appreciated by human players, significant changes in opponent behaviour should not take a considerable number of generations. Therefore, the population should be subjected to significant changes in every new generation. As variation operators we are given the choice between a genetic recombination of genes, a genetic mutation of genes, or both.

A recombination of genes results in erratic variation of the genetic information. In our real-time environment we are evolving genomes online and therefore can only evaluate one genome at a time. Erratic variation of genetic information is likely to result in undesirable behaviour. Consequently, the online evolutionary mechanism should not allow these genomes to be introduced in the population. Thus, a genetic recombination of genes is not suitable in our environment.

The only suitable variation operator is a genetic mutation. We choose for a so-called scaled mutation operator. The general idea behind a scaled mutation comes from the desire of giving the mutation operator a less static nature. Instead of always having the same absolute mutation, a scaled mutation operates by mutating in close relation with the achieved fitness value of a genome. We distinguish two different types of genome-classes:

1. **Genomes which obtained a low fitness.** These genomes are subdued to a large mutation since they do not seem able to achieve (satisfactory) progress.

2. **Genomes which obtained a high fitness.** These genomes are subdued to a small mutation since they most likely approach an optimum.

3.4.6 Survivor selection

We defined a small finite-sized population, which implies that we have to make a choice on which individuals we are going to allow in the new generation. An important consideration for the design of this survivor selection is providing genetic diversity in the population, thus protecting the evolutionary approximation against becoming too greedy and getting stuck in a local optimum. We decided not to reseed the next generation with genomes based on selected children, but provide genetic diversity in the population by using a form of

elitism. Our population is used as a storage location for preservation of best-so-far genomes.

Since the environment we are using is time-critical we desire the fitness value of the genomes to constantly improve. Therefore, we designed the survivor selection mechanism to replace the worst genome with a better genome. If an inferior genome is offered to the population, the genome is discarded from the population.

3.5 Fail-safe considerations

Let us start this section with an appropriate quote from Frank Herbert [21]:

> *"A beginning is the time for taking the most delicate care that the balances are correct."*

This quote describes the characteristic risks of evolutionary algorithms in a nutshell. By definition an evolutionary algorithm is a shining example of so-called black-box technology, once it is running we can observe its effects but have no direct control on the process anymore. The lack of control can cause a population to evolve to an undesirable direction. However, if we take a look at biology, human physiology provides means of influencing the quality of offspring. For instance, eminent abnormalities of an embryo that can develop towards a non-viable fetus can cause a naturally induced abortion of the embryo [51]. Another example, the fertility of pre-menopausal females has typically been significantly reduced [22]. Both examples are contributed to a natural mechanism of ensuring the quality of the offspring, thus indirectly having a positive effect on the course of evolution.

Fail-safe considerations in evolutionary algorithms are required, since we desire means of controlling the evolution. TEAM explicitly needs a mechanism to ensure the quality of the population since it only occasionally can evaluate genomes and since the size of the population is small.

This section describes two such mechanisms we designed for TEAM. In section 3.5.1 we discuss the fitness recalculation of genomes in the population. In section 3.5.2 we discuss history fall-back, a concept for allowing the evolution to revert to a previous state.

3.5.1 Fitness-recalculation mechanism

In commercial computer games we typically have to deal with a large amount of randomness. Specifically in the action-game genre, this randomness, or chance as some would call it, is very typical. Often players describe it as simply being in the wrong place at the wrong time. For instance, in Quake III, unexpected encounters with opponents or traps can result in getting killed before one even gets the chance to start a tactical manoeuvre.

Randomness in the environment poses a major problem for evolutionary algorithms. Unlike discrete games with perfect information, it is difficult to

demarcate between our opponent being unlucky and our team using an effective strategy. This poses a problem for our evaluation function since we cannot be sure the fitness is the direct result of the tactic expressed by the genome.

TEAM implements a solution to this uncertainty based on a straightforward observation. Imagine a genome that is used in a specific state obtains a high fitness. Assume we would select this genome for the next cycle of the evolutionary process. The selected genome obtained a high fitness and shall therefore undergo only a slight mutation. If the original genome really was 'good', this new genome obtains a comparable fitness. But, if the high fitness of the previous genome was caused by lucky circumstances, those circumstances presumably no longer exist, and as a result the fitness of the new genome would be significantly lower in comparison with the parent genome.

Taking the observation, described in the previous paragraph, into account solves the demarcation problem. TEAM recalculates the fitness of all parent genomes dependant on the fitness of their children. Should the children of a 'good' genome obtain low fitness values then the fitness of the parent genome gradually decrease. Since the fitness of the parent decreases, eventually this 'bad' parent is removed from the population.

The recalculation process serves as a filter for only keeping genomes in the population that are genuinely fit. This security measure restrains the evolutionary algorithm from using inferior genomes as a basis for further evolution.

3.5.2 History fall-back concept

In our environment we need the evolutionary mechanism to be able to revert to an earlier state. In theories concerning natural evolution this demand for reversibility would be a contradicting one, since by definition the process of evolution cannot be reversed. In our environment however it is a necessity.

Let us illustrate the necessity of reversible evolution by an example. Normally one is evolving towards a termination condition based on reaching a known optimum of the designed evaluation-function. Due to the stochastic nature of evolutionary algorithms, such a condition may not exist or may never be reached, therefore a need for extending the stop criteria with secondary conditions arises. Conceptually the idea remains that a population evolves in a certain direction. In each cycle of the evolutionary algorithm the population alters to a more fit population. Now, in our environment the fitness of the population is based on how it is performing, and this performance indirectly depends on our opponent team. Assume that after several dozen of cycles we have evolved a population that is dominant over the opponent team. Since the population has learned a better strategy, one is inclined to believe that we have reached an optimum that satisfies a stop criterion.

Assume that in this moment the opponent devises a strategy that is superior to the one we are currently using. The very strategy we once considered optimal can be inferior in changing circumstances. Typical evolutionary algorithms are not designed to deal with such changes. If after a number of generations a population approaches an optimum, the population no longer has the genetic

diversity needed to evolve to a new optimum when the optimum condition itself shifts.

TEAM is capable of dealing with little genetic diversity in the population. We designed the adaptive mechanism to create diversity in the next population based on best-so-far genomes in the population, in combination with a well-scaled mutation. Should the evolution be 'stuck' somewhere it can always fall back on best-so-far genomes. One can correctly note that there is a possibility of the population containing only over-fitted genomes, which is unsuitable in changing circumstances. Yet, child genomes of this over-fitted population obtain a low fitness in next cycles of the evolutionary algorithm. Consequently, the fitness-recalculation mechanism ensures that inferior genomes are removed from the population if the fitness of their children remains low. Additionally, the population is updated with genomes that have been subdued to a large mutation, thus restoring the genetic diversity in the population.

We expect the history fall-back mechanism to provide the ability to cope with changing opponent behaviour. In section 4.3 we discuss an experiment which analyses the performance of TEAM in an environment with changing opponent behaviour.

3.6 Implementation of TEAM

This section discusses the implementation of TEAM, in accordance with the second research objective.

The design of TEAM is implemented in the game Quake III. ID Software, the creator of Quake III, made available the latest version of the game source code. With the game source-code, a modified Quake III release was built. A modification (or 'mod') is essentially a new game one can play in the Quake III environment, using modified rules, weapons, levels, AI etc. ID Software has always allowed the gaming community to create their own modifications, which is likely to be the main reason for the longevity and dedicated community that surrounds the Quake-family of games. To give an impression of the size of the current community, in the year 2003 hundreds of Quake III servers are online on a daily basis, and modifications are released regularly [31]. Creating a modification for the Quake III environment offers the advantage of programming C++ in an industrial standard IDE. Quake III modifications are compiled in the form of a so-called Quake Virtual Machine, which is a safe and platform independent alternative for .dll files.

We implemented the design of TEAM in the form of a modification. The modification allowed a team of agents, controlled by the Quake III team AI, to play against a team of agents controlled by TEAM. Note that we left the original code intact, TEAM is provided as an option for the old team-oriented artificial intelligence. In figure 3.5 we present a global representation of the implementation of TEAM in the Quake III environment. TEAM is called by a state-transition event in the Quake III environment. Subsequently, the adaptive mechanism outputs a genome, which is processed by the Quake III environment.

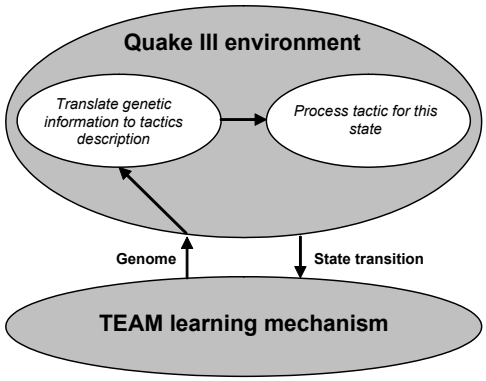


Figure 3.5: Implementation of TEAM in the Quake III environment.

3.7 Summary

This chapter discussed the first research objective; designing a mechanism that imposes adaptive team AI on opponents in commercial computer games. Additionally, the chapter discussed the second research objective; implementing the design in a test environment.

The chapter presented the design of TEAM. We described evolutionary algorithms, which are our main source of inspiration for designing TEAM. Based on evolutionary algorithms, we described two possible approaches for a team-oriented adaptive mechanism. First, a centralised adaptive mechanism approach, and second, a decentralised adaptive mechanism approach.

Founded on the design choice of designing an centralised team-oriented adaptive mechanism, we gave an in-depth description of TEAM. TEAM employs four features, which distinguish the adaptive mechanism from typical evolutionary approaches. First, a centralised agent control mechanism evolution. Second, a mutualistic-symbiotic evolution. Third, a state-specific genome evolution. Fouth, evolution with history fall-back.

The uncontrollable nature of evolutionary algorithms poses a risk to our research since we want to assure that the algorithm evolves within devised boundaries. We therefore discussed two fail-safe considerations for the adaptive mechanism. First, we discussed recalculating the fitness of parent genomes based on the fitness of their children. Second, we discussed the ability of the evolution to revert to an earlier state.

In addition to the design of TEAM, we discussed the implementation of TEAM in the form of a modification for Quake III, a typical commercial computer-game.

Chapter 4

Experiments

This chapter discusses performing experiments, preparatory to discussing the third research objective; obtaining a qualitatively acceptable performance of the adaptive mechanism.

The experiments concerned testing the capability of TEAM to successfully learn while competing against static team AI (experiment 1), and the capability of TEAM to successfully learn while competing against the Quake III team AI (experiment 2). Before these experiments are discussed in detail, the evaluation of an experimental run is discussed in section 4.1. Subsequently, the first experiment is discussed in section 4.2, and the second experiment in section 4.3. Finally, a summary of the chapter is given in section 4.4.

4.1 Evaluation of an experimental run

An experimental run consists of two Quake III teams playing the Capture-The-Flag team-based game mode until the game is interrupted by the experimenter. In an experimental run, a team that is controlled by TEAM is competing against a team with non-adaptive team AI. To quantify the performance of TEAM, two properties of an experimental run are used: the absolute turning point, and the relative turning point.

First, we define the absolute turning point as the first moment in which the team using TEAM statistically outperformed the other team. To test the hypothesis that the team controlled by TEAM outperformed the other team, we observe the sampling distribution of the proportion p under the null hypothesis that both teams are of equal strength, and the alternative hypothesis that TEAM outperformed the other team. The sampling distribution of p is a discrete distribution of two parameters: (1) N, the number of samples, and (2) r, the probability that the performance of both teams is equal. Assuming the performance of both teams is equal ($r = .5$), the probability of the sample result p is displayed in expression 4.1, where the counter i starts with the number of wins w for the adaptive opponent. Using a sample size of $N = 20$ we denote the

Win:Lose ratio	Chance of unequal performance
12 : 8	74.83%
13 : 7	86.84%
14 : 6	94.23%
15 : 5	97.93%
16 : 4	99.41%

Table 4.1: Chance of TEAM outperforming the opponent.

chance of TEAM outperforming the opponent in table 5.1. Observe that the chance for unequal performance at a ratio of 15 wins against 5 losses is 97,93%. When this ratio is achieved, we reject the null hyphothesis. We expect this ratio to be feasible for TEAM. Therefore, we redefine the absolute turning point as the first moment in which the team controlled by TEAM obtains a win:loss ratio of a least 15 wins against 5 losses in a sliding window of $N = 20$. Note that therefore 20 is the lowest absolute turning point we can reliably calculate.

$$p = \sum_{i=w}^{N} \frac{N!}{i!(N-i)!}(.5)^N \tag{4.1}$$

Second, we quantify the noticeable effect of TEAM by defining the relative turning point as the last moment in which the team using TEAM has a zero lead with respect to the other team, with the requirement that from this moment on the team using TEAM does not lose its lead for the rest of the experimental run.

4.2 TEAM vs. Static team AI

This section discusses the experiment where TEAM is competing against static team AI. The focus of the experiment is to confirm that a team controlled by TEAM can successfully adapt. In sub-section 4.2.1, the experimental setup is discussed. Sub-section 4.2.2 discusses the results of the experiment, and sub-section 4.2.3 draws a conclusion from the experimental results.

4.2.1 Experimental setup

In an experimental run two Quake III teams play the Capture-The-Flag team-based game mode until the game is interrupted by the experimenter. One team is controlled by TEAM, while the other team is controlled by static team AI. Combat between teams takes place in a so-called open map, of which an example is displayed in figure 4.1. In an open map there are no walls. Therefore, agents have an unrestricted view over the environment. Thus, open maps minimize the chance for coincidental agent-encounters. The individual agent AI, means of communication, and team organisation of both team's agents is the same. Both

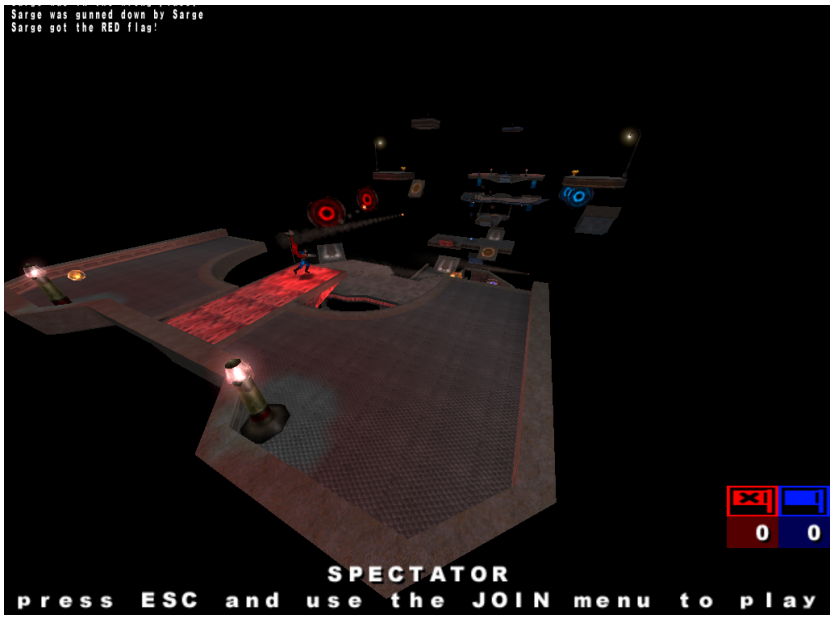


Figure 4.1: Example of agents operating in an open map.

teams, each consisting of four agents, only differ in the used control mechanism; one team uses an adaptive control mechanism (TEAM), while the other team uses a non-adaptive control mechanism (static team AI). By merely laying our focus on the control mechanism we are able to justly draw conclusions concerning the performance of TEAM, given the experimental-results. We expect the team of agents controlled by TEAM to discover a way to exploit static behaviour of the static team AI.

The static team is implemented by temporarily modifying the code of the original team AI in the Quake III environment in such way that it results in static team behaviour. The team using TEAM is initialised with the same behaviour as the static team.

We first performed a long experimental run to obtain a general impression of TEAM's performance. Next we performed five short experimental runs. The long experimental run is a run we let continue over a twenty-hour period. Short experimental runs are runs that were typically executed during a six-hour period. In a short experimental run typically a total of about 200 points is scored.

4.2.2 Results

In this sub-section, we discuss two experimental results: (1) the results of the long experimental run, and (2) the results of the short experimental runs.

First, in the long experimental run the sum of the points scored by both

Experiment #	Absolute turning point	Relative turning point
1	63	8
2	150	144
3	70	52
4	137	126
5	173	12

Table 4.2: TEAM vs Static team AI - Turning points.

teams is 692. Of this total, the team controlled by TEAM obtained 520 points, and the team controlled by the static team AI obtained the remaining 172 points. This obviously shows that the adaptive team has become significantly better than the static team. The course of the absolute performance of TEAM is plotted in figure 4.2, additionally with a sixth-order polynomial trendline. The x-axis denotes the number of points that are scored by both teams. The value on the y-axis denotes the number of points which the team using TEAM scored over the last twenty scored points. Thus, if both teams are of equal strength, one observes an y-axis value of 10. The course of the relative performance of TEAM is plotted in figure 4.3. The x-axis denotes the number of points that are scored by both teams. The value on the y-axis denotes the lead of the adaptive team with regard to the non-adaptive team. The absolute turning point of this experimental run is 72, while the relative turning point is 62.

Second, short experimental runs showed an equivalent course, compared to the long experimental run. The course of the absolute and relative performance of a typical short experimental run is displayed in figure 4.4 and 4.5, respectively. In this experimental run the sum of the points scored by both teams was 175, of which the team controlled by TEAM obtained 116 points, and the team controlled by the static team AI obtained the remaining 59 points. Like the result of the long experimental run, this result obviously shows that the adaptive team has become significantly better than the static team. The turning points of each experimental run are displayed in table 5.2. A full listing of all test results is given in appendix A.

4.2.3 Conclusion of the results

In each experimental run we observe that initially the performance of both teams is similar. In time, and without any significant degradation in the adaptive-team's performance, the lead of the adaptive team sharply increases. From this result, we may draw the conclusion that TEAM is capable of successfully adapting to static opponent behaviour.

Because the experiment against the static team AI was focussed on confirming that a team controlled by TEAM can successfully learn, which all experimental runs did, we decided to forego further tests against static team AI and tackle the more difficult task of pitting TEAM against the original Quake III

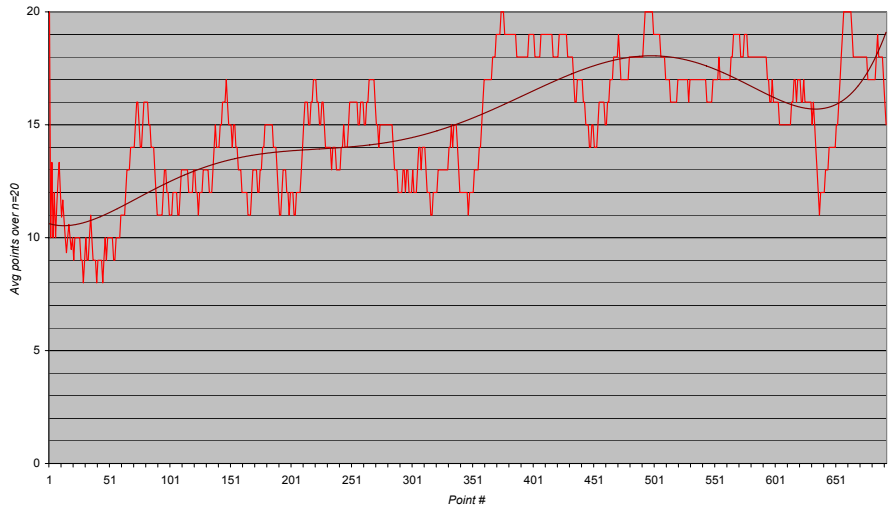


Figure 4.2: TEAM vs Static team AI - Long Run (Absolute performance).

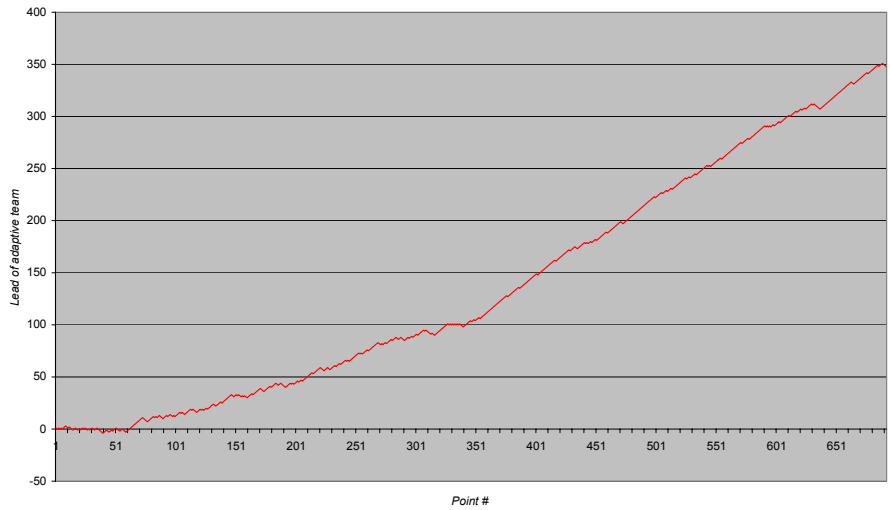


Figure 4.3: TEAM vs Static team AI - Long Run (Relative performance).

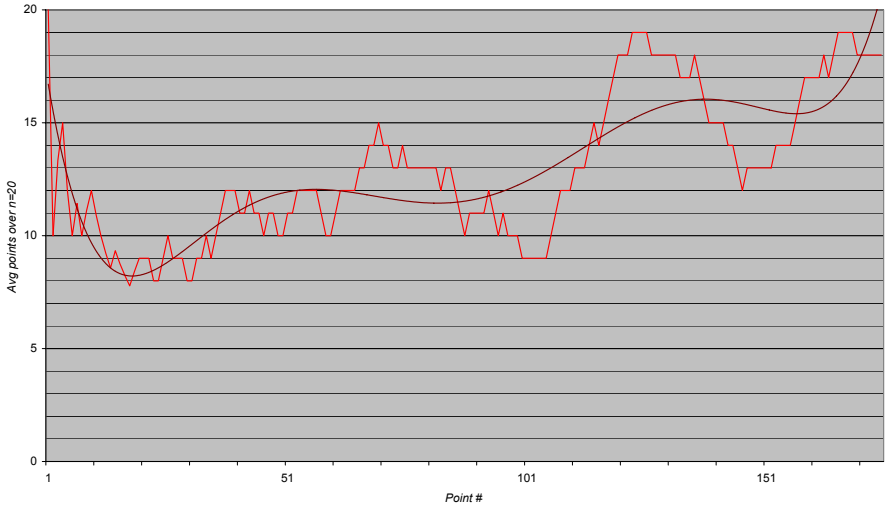


Figure 4.4: TEAM vs. Static team AI - Short Run (Absolute performance).

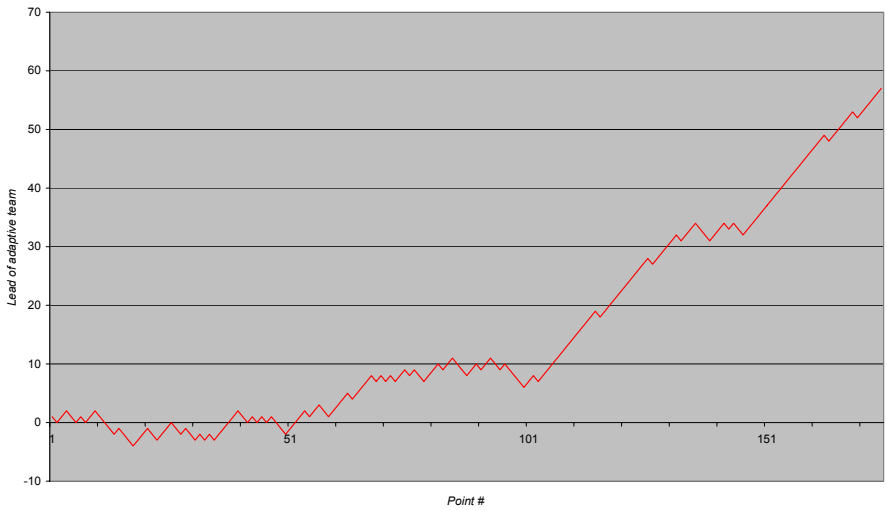


Figure 4.5: TEAM vs. Static team AI - Short Run (Relative performance).

team AI.

4.3 TEAM vs. Quake III team AI

This section discusses the experiment where TEAM is competing against the Quake III team AI. The focus of the experiment is to confirm that a team controlled by TEAM can successfully adapt in an environment where the opponent behaviour changes. In sub-section 4.3.1, the experimental setup is discussed. Sub-section 4.3.2 discusses the results of the experiment, and sub-section 4.3.3 draws a conclusion from the experimental results.

4.3.1 Experimental setup

The setup of this experiment is similar to the previous experiment, with the difference that TEAM is tested against a dynamic team AI, instead of static team AI. For this purpose we pitted TEAM against the original Quake III team AI, which uses intelligent switching between static tactics. In comparison to static team AI, the team AI of Quake III provides more dynamic team-behaviour. It is dynamic in a sense that the team is capable of efficiently using different tactics based on the state of the Capture-The-Flag team-based game mode, and has resemblance with how human players operate in team-oriented environments. In Quake III, a tactic is a division of roles description linked with a specific state.

In Quake III the dynamic team AI is implemented in the following fashion. Normally a team is using tactic x. However, a team switches to tactic y in case it is not able to capture a flag within a preset period of time. Tactic y implements a more aggressive style of playing, where tactic x is more defensive.

Competing against a dynamic team is more difficult since winning the game is no longer restricted to finding tactics that are dominant over merely one static tactic. Dominant tactics for competing against a static opponent are likely to be insufficient in changing circumstances. Note that the circumstances change, it is a certainty that the opponent team changes its behaviour if the adaptive team wins. Even though the Quake III team AI changes between merely two tactics, the behavioural shift poses a challenge for TEAM since it now has to deal with significant behavioural changes of the opponent.

However, TEAM is designed to be capable of defying changes in the opponent behaviour. It therefore is our expectation that dominant behaviour occurs in due time. The test results discuss if our expectation is justified.

4.3.2 Results

This sub-section discusses the results of a team controlled by TEAM competing against a team controlled by the original Quake III team AI. In the first experimental run, a total of 700 points are scored, of which the team controlled by TEAM obtained 497 points, and the team controlled by the original Quake

Experiment #	Absolute turning point	Relative turning point
1	148	20
2	263	158
3	106	70
4	38	36
5	99	50
6	42	24
7	58	114
8	107	48
9	205	132
10	92	32
11	61	56
12	127	144
13	136	50
14	91	82
15	53	54

Table 4.3: TEAM vs Quake III team AI - Turning points

III team AI obtained the remaining 203 points. This obviously shows that the adaptive team has become significantly better than the team controlled by the original Quake III team AI.

To determine the reproducibility of the first experimental run, the test-match is repeated several times. The turning points of each experimental run are presented in table 5.3. A full listing of all test results is given in appendix B. To give an impression of the course of a typical experimental run, we plotted the absolute performance in figure 4.6. Additionally, the course of the relative performance of this experimental run is plotted in figure 4.7.

The *average* of all absolute turning points is 108.40, with a *standard deviation of the mean* of 61.99. Note that the *median* of these results is 99. The *maximum* observed absolute turning point is 263, while the *minimum* absolute turning point is 38. Additionally, the *standard error of the mean* is 18.69, which indicates that with 68.27% certainty the real average of the absolute turning point lies in the interval $[90, 127]$.

Regarding the relative turning point, the *average* of all relative turning points is 71.33, with a *standard deviation of the mean* of 44.78. The *median* of these results is 50. Additionally, the *maximum* observed relative turning point is 158, while the *minimum* relative turning point is 20. The *standard error of the mean* is 13.50, which indicates that with 68.27% certainty the real average of the relative turning point lies in the interval $[58, 85]$.

4.3.3 Conclusion of the results

In each experimental run we observed that initially the performance of both teams is similar. In time, and without any significant degradation in the adaptive team's performance, the lead of the adaptive team sharply increases. In all experimental runs, TEAM learned to significantly outperform the Quake III team AI. From this result we may draw the conclusion that TEAM is capable of successfully adapting to significant changes in the opponent behaviour.

4.4 Summary

This chapter discussed performing experiments, in order to draw a conclusion with regard to the third research objective; obtaining a qualitatively acceptable performance of the adaptive mechanism. A qualitatively acceptable performance is denoted by the capability of TEAM to successfully learn while competing against static team AI (experiment 1), and the capability of TEAM to successfully learn while competing against the Quake III team AI (experiment 2).

The first experiment concerned testing TEAM competing against static team AI. From the experimental results we may draw the conclusion that TEAM is capable of successfully adapting to static opponent behaviour.

The second experiment concerned testing TEAM competing against the Quake III team AI. From the experimental result we may draw the conclusion that TEAM is capable of successfully adapting to changes in the opponent behaviour.

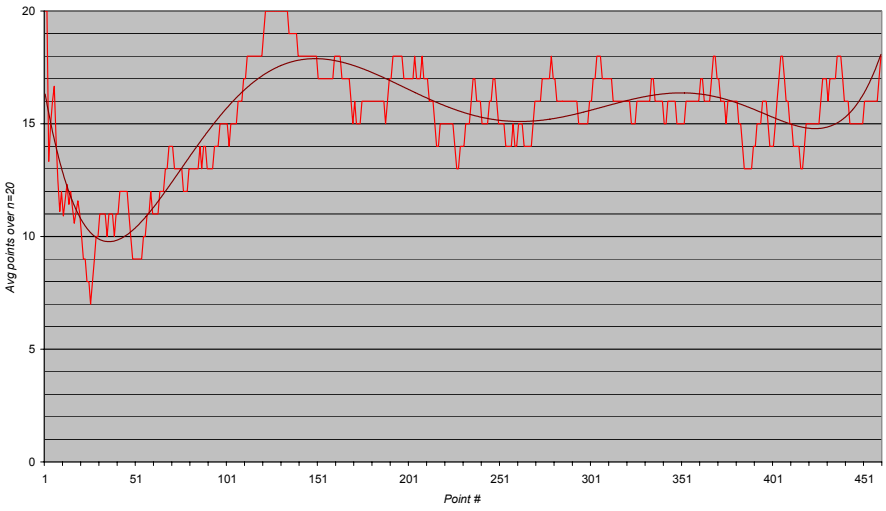


Figure 4.6: TEAM vs. Quake III team AI - Typical Run (Absolute performance).

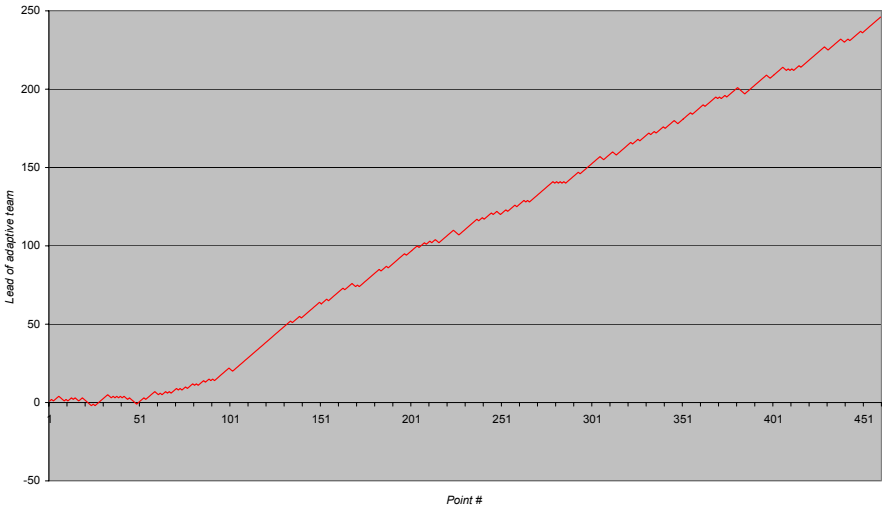


Figure 4.7: TEAM vs. Quake III team AI - Typical Run (Relative Performance).

Chapter 5

Discussion

In this chapter four topics are discussed. First, in regard to the third research objective; obtaining a qualitatively acceptable performance of the adaptive mechanism, section 5.1 evaluates the qualitative performance of TEAM. Second, in section 5.2 the behaviour learned by the TEAM is discussed. Third, section 5.3 discusses the phenomenon of TEAM 'forgetting' tactics. Fourth, section 5.4 discusses the relationship between artificial intelligence and entertainment in commercial computer games.

5.1 Qualitative evaluation of TEAM

In this section we evaluate the qualitative performance of the adaptive mechanism. TEAM is an online adaptive mechanism, and for online adaptation to work in practice, we denote four requirements for qualitative performance. First, it must be computationally fast. Second, robust with respect to randomness inherent in the environment. Third, efficient with respect to the number of adaptation trials, and fourth, effective with respect to the intermediate AI generated during the adaption phase [38]. Below we discuss each of these four requirements in detail.

1. **Computationally fast.** In our environment the adaptation-process takes place online. Therefore the adaptation mechanism should only consume a minimum of processing resources to prevent disruption of the flow of the gameplay. TEAM is computationally fast because it only needs to return a single genome to the application and update a small population after each state-transition.

2. **Robust.** TEAM is robust in a sense that while the gaming mechanism of action-games is largely probabilistic, it is able to cope with a large amount of randomness in the environment. As already discussed in chapter 3, TEAM uses a delayed-reward mechanism (sub-section 3.4.4), a fitness-

recalculation mechanism (sub-section 3.5.1), and a history fall-back concept (sub-section 3.5.2) to impose robust behaviour.

3. **Efficient.** TEAM is efficient with respect to the number of trials required for a human player to notice the effects of adaptation of opponent's behaviour. In chapter 4 we stated that the average relative turning point of a team controlled by TEAM is 71.33. We are particularly pleased with this result considering that TEAM has to learn against the original Quake III opponent who already behaves efficiently, and by adaptation of agent behaviour significantly outperforms this opponent. Should the opponent we tested TEAM against be not as fine-tuned as the original Quake III opponent, the average relative turning point would have been even lower.

4. **Effective.** The effectiveness of an adaptive mechanism is measured in terms of the challenge adaptive agents pose compared to e.g., non-adaptive agents. Evolutionary algorithms are often disregarded from research since they generally are in-effective for online adaptation tasks [38]. Yet, observing the experimental results it is a great gratification that TEAM, which is inspired by evolutionary algorithms, is indisputably effective considering the fact that it outperformed non-adaptive opponents without any significant degradation in performance.

In summary, we may conclude that TEAM is computationally fast, robust, efficient and effective.

5.2 Learned behaviour

Analysing the behaviour of TEAM, we observed that the population does not converge to a single dominant strategy. On the contrary, TEAM is continuously adapting to the environment in order to remain dominant on an overall basis.

In this section we discuss towards which tactics TEAM is inclined to evolve. We observed that in state 'Both flags at their base', the adaptive team tends to go on full offence. Additionally, in state 'Base flag stolen', it tends to become predominantly offensive, and in state 'Enemy flag stolen' it tends to become predominantly defensive. Finally, in state 'Both flags stolen' the adaptive team tends to become predominantly offensive.

Remarkably, the team has learned to go on full offence when both flags are at their base. Considering that the team risks losing its own flag due to an inferior defence, a qualitative explanation of the benefit of this tactic is in place.

A primary requirement for returning the enemy flag to the friendly base, and thus scoring a point, is first actually capturing the enemy flag. The adaptive team learned to go on full offence in state 'Both flags at their base'. Yet, the non-adaptive team's tactic is moderate in this state; some agents are assigned to an offensive role, while others are assigned to a defensive role. Due to the offensive field supremacy of the adaptive team in this state, it is plausible that the adaptive team manages to capture the enemy flag.

Still, in order for the adaptive team to score a point, it needs to return the flag to the friendly base. Constant adaptation of the other states ensures that tactics are used which provide a balance between on the one hand defending the agent that is trying to return the enemy flag to the friendly base, and on the other hand preventing the enemy team from capturing and returning the friendly flag to its base. As can be intuitively felt, ensuring that the enemy flag is captured, and successively retaining this positive momentum, is an effective way to win the Capture-The-Flag team-based game mode.

The original Quake III team AI uses only moderate tactics in all states. Therefore, it is not able to counter any field supremacy. This exemplifies the inadequacy of non-adaptive artificial intelligence. Despite the fact that the original Quake III team AI is fine-tuned to be suitable for typical situations, it cannot adapt to superior player tactics, like the above-sketched situation.

5.3 Forgetting tactics

This section discusses the phenomenon of TEAM 'forgetting' tactics, presents an analysis of its occurrence, and suggests how to prevent the occurrence of this phenomenon.

We noticed that if we let the experiments continue even after an absolute turning point was discovered, it sometimes happened that the quality of the population started to degrade, which resulted in inferior behaviour. An experimental run where this phenomenon occurs is displayed in appendix B.4. In most adaptive mechanisms this phenomenon occurs because the concerning mechanism continues to learn new behaviour, even when it is already successful. In practice the effects of continuously adapting are often informally described as if the adaptive mechanism is 'forgetting' what it previously learned. As noted in [38], simply stopping the adaptive mechanism when it has reached an optimum is not a good solution, because the agents should still be able to adapt to changing player tactics. We therefore implemented a fitness-recalculation mechanism, which was already discussed in section 3.5.1, to protect the population from degrading.

However, in the experimental run displayed in appendix B.4, we observed that under special circumstances it is possible that the quality of the population degrades. Ironically, in contrast to the fitness-recalculation mechanism's design goal of preventing the quality of the population to degrade, the fitness-recalculation mechanism is sometimes obliquely causing the degradation. Recall that we continuously recalculate the fitness value of parent genomes in the population. Assume that the population consists of genomes of a relatively high quality, but that the children of these genomes obtain low fitness values due to some unlucky circumstances. Consequently, the fitness value of the 'good' parent genomes decreases. Since in this case the fitness value of some parent genomes in the population is relatively low, it is likely that they are replaced by genomes which receive higher fitness values in the present 'unlucky' circumstances. However, such genomes clearly got lucky when they were evaluated,

and consequently their fitness value decreases quickly after being added to the population. Thus, the quality of the population sometimes is inclined to decrease, because genomes which seem superior (but are not) can replace genomes which are superior (but don't seem to be).

A potentially significant degradation of the population's quality is caused by the replacement of the complete population by inferior genomes. An example of this phenomenon is displayed in appendix C. Note that the adaptive mechanism still is able to recover from the behaviour caused by such an inferior population, but fact is that the population is designed for preservation of best-so-far genomes and therefore should not be easily replaced as a whole.

To prevent a degrading quality of the population therefore implies that genomes in the population are replaced if, and only if, a higher certitude of their 'good' performance can be obtained. Such increased safekeeping of the population entails that changes in the population are enforced at a slower rate. Therefore, in commercial application of TEAM, a balance should be found between on one hand sustaining earlier behaviour, and on the other hand quickly responding to behavioural changes.

5.4 AI and entertainment

TEAM is specifically designed to create a strong AI system capable of defeating the best human players, as is common practice in academic AI game-research [8]. However, game developers do not share the academia's objective of creating strong AI systems. With regard to artificial intelligence, their focus comprises maximizing the entertainment value of their products [8]. Imagine that a game is published where human players cannot defeat the artificial opponents. This would not be entertaining; rather, it would be a frustrating experience for human players. For instance, Quake III agents could easily shoot with 100% accuracy, but in practice they deliberately do not.

Regarding the effectiveness of TEAM, we designed the adaptive mechanism to increase its performance by adaptation based on the assumption that a strong human player sees this adaptive behaviour as a challenge. However, assume that we are not dealing with a strong human player, but with a novice player, then evidently this person is not entertained if the game is too difficult.

Therefore, strong AI systems have to be toned down to adjust to human players in such a way that human players are continuously challenged, instead of being 'sadistically slaughtered'. For instance, in the action game genre, computer-game opponents should strive to be only slightly superior to human players. Thus, these computer-game opponents are challenging enough for human players to continuously attempt to win, and at the same time, are not frustrating since they no longer unremittingly win significantly.

Concluding, for entertainment purposes, the adaptive mechanism should not be focussed on winning, but rather should be based on subjective criteria which ensure that a challenging 'balance' between artificial opponents and human players emerges. With TEAM this can be possibly accomplished by redesigning the

evaluation function. Future research should determine how well this works.

Chapter 6

Conclusions and future research

At the end of this thesis we return to the problem statement and research question. Section 6.1 revisits the research question. In section 6.2 the problem statement is answered. Subsequently, section 6.3, describes future research directions.

6.1 Answer to the Research Question

In section 1.3 we presented our problem statement and posed one research question that should be answered before we could deal with the problem statement. In this section we answer the research question. In the next section we will formulate from this answer a reply to the problem statement.

Our research question was:

> **Research question:** *Is it possible to create a mechanism that imposes adaptive team AI on commercial computer-game opponents and achieves a qualitatively acceptable performance?*

In our attempt to answer the research question, we have three objectives.

1. Designing a mechanism that imposes adaptive team AI on opponents in commercial computer games.
2. Implementing the design in a test environment.
3. Obtaining a qualitatively acceptable performance of the adaptive mechanism, i.e., performance that is computationally fast, robust, efficient and effective [38].

In accordance with the first research objective, we designed a team-oriented adaptive mechanism and named it the *Tactics Evolutionary Adaptability Mechanism* (TEAM).

In accordance with the second research objective, we successfully implemented the design of TEAM in Quake III, a state-of-the-art commercial computer-game.

In accordance with the third research objective we evaluated the adaptive capability of TEAM. We denoted two experiments. The first experiment concerned testing TEAM while competing against static team AI, and was aimed at investigating the adaptive capability of TEAM. The second experiment concerned testing TEAM while competing against the original Quake III team AI, and was aimed at investigating the adaptive capability of TEAM in an environment where the opponent behaviour changes. Based on the experimental results, discussed in chapter 4, we drew the following conclusions:

- TEAM is capable of successfully adapting to static opponent behaviour.
- TEAM is capable of successfully adapting to changes in the opponent behaviour.

The conclusions of both experiments show that TEAM endows opponents in team-oriented commercial computer games with successful adaptive behaviour.

We evaluated the requirements for qualitatively acceptable performance [38], and drew the conclusion that TEAM is computationally fast, robust, efficient and effective. Thereupon, we may draw the conclusion that TEAM obtained a qualitatively acceptable performance.

By achieving all our research objectives, we may draw a final conclusion by answering the research question with an unequivocal yes, it is indeed possible to create a mechanism that imposes adaptive team AI on commercial computer-game opponents and achieves a qualitatively acceptable performance.

6.2 Answer to the Problem Statement

Our problem statement was:

> **Problem statement:** *Is it possible to improve the performance of opponents in state-of-the-art commercial computer games, with regard to their team-oriented behaviour?*

Our approach to answering the problem statement was aimed at creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment. Taking the answer to the research question above into consideration, the answer to the problem statement must be that it is indeed possible to improve the performance of opponents in state-of-the-art commercial computer games, with regard to their team-oriented behaviour.

The answer to the problem statement does not preclude that there are alternative ways to improve the performance of opponents in state-of-the-art commercial computer games, and even further improve upon adaptive team-oriented behaviour.

TEAM is capable of unsupervised and intelligently adapting to the environment, and as stated in section 5.2, TEAM adapted to the environment in such a way that it evolved towards dominant tactics. These dominant tactics, which resulted in 'dangerous' but successful behaviour, exceeded the vision of the designers of Quake III, which was focussed at always behaving moderately. Therefore, we have fulfilled our aim of creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment.

In chapter 1 we stated that adaptive team AI in commercial computer games does not exist. Therefore, our results provided a significant contribution to the study and application of artificial intelligence techniques in general, and machine-learning techniques in commercial computer games in particular.

6.3 Recommendations for Future Research

In chapter 5 we discussed that TEAM is suitable for online team-oriented artificial intelligence of commercial computer-game opponents. However, some more research must be done before TEAM can be used for entertainment purposes in commercial computer games. We specifically demarcate between on the one hand designing an effective adaptive mechanism, and on the other hand designing an entertaining adaptive mechanism. Designing an effective adaptive mechanism typically is the concern of computer science research, where an entertaining adaptive mechanism is an interesting area of research for psychologists and cultural-scientists. Therefore, the topic of entertainment poses a suitable area of future research, particularly with an eye on commercial implementation of machine-learning techniques in commercial computer games.

On a personal note, we suggest extending the research to highly tactical team-oriented commercial computer games, such as games in the real-time strategy game genre. As Buro [8] states, unlike other game genres, these games offer a large variety of fundamental AI research problems, like adversarial real-time planning, decision making under uncertainty, spatial and temporal reasoning, pathfinding, resource management and collaboration, learning, and, opponent modeling. We therefore endorse the vision of Buro that the results of real-time strategy game research increases our understanding of fundamental AI problems, like the ones listed above, and has considerable impact on the real-time control domain in general and the computer games industry in particular, which is in need of creating intelligent commercial computer-game opponents.

The design of TEAM positively answered the problem statement, by successfully adapting tactics. However, we recommend future research to be, ultimately, aimed at creating adaptive AI capable of deciding which behavioural adaptations are required, and determine how to best accomplish the adaptation.

References

[1] Adams, D. (1995)
"The Original Hitchhiker Radio Scripts".
Harmony Books
ISBN: 0517883848

[2] Aha, D. W. (1991)
"Case-based learning algorithms".
In Proceedings of the DARPA Case-Based Reasoning Workshop, pp. 147–158
Washington, D.C.: Morgan Kaufmann

[3] Bachmann, F. et al. (2000)
"Software Architecture Documentation in Practice: Documenting Architectural Layers".
SPECIAL REPORT CMU/SEI-2000-SR-004 - March 2000
Carnegie Mellon - Software Engineering Institute

[4] Barnes, P. and Hutchens, J. (2002)
"Testing Undefined Behavior as a Result of Learning".
AI Game Programming Wisdom (ed. Rabin, S.), pp. 615-623
Charles River Media

[5] Boullart, L., Krijgsman, A. and Vingerhoeds, R.A. (1992)
"Application of artificial intelligence in process control".
Pergamon Press
ISBN: 0080420176

[6] Buche, C., Parenthoën, M., Tisseau, J. (2002)
"Learning by imitation of behaviour for autonomous agents".
Laboratoire d'Informatique Industrielle, ENIB
Université de Bretagne Occidentale, France

[7] Buijs, A. (1999)
"Statistiek om mee te werken".
Zesde druk, derde oplage, Educative Partners Nederland
ISBN: 90-207-2733-8

[8] Buro, M. (2003)
"RTS Games as Test-Bed for Real-Time AI Research".
Department of Computing Science, University of Alberta, Canada
Proceedings of the 7th Joint Conference on Information Science, JCIS 2003,
Editors: Chen, K., et al.

[9] Clarke, A.C. (1972)
"Profiles of the Future".
Indigo, London

[10] Cohen, P.R. (1995)
"Empirical Methods for Artificial Intelligence".
ISBN: 0-262-03225-2 (HC)

[11] Darwin, C. (1859)
"The Origin of Species, by means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life".
Murray of London, London, United Kingdom

[12] Demasi, P., Cruz, A.J. de. O. (2002)
"Online coevolution for action games".
Instituto de Matemática - Ńucleo de Computação Electrônica
Universidade Federal do Rio de Janeiro

[13] Dijkstra, E.W. (1959)
"A note on two problems in connection with graphs".
Numeriche Math, 1, pp. 269-271

[14] Donkers, H.H.L.M. (2003)
"Nosce Hostem, Searching with opponent models".
SIKS Dissertation Series No. 2003-13
Universiteit Maastricht, The Netherlands
ISBN: 90-5278-390-X

[15] Duan, J., Gough, N.E., Mehdi, Q.H. (2002)
"Multi-agent reinforcement learning for computer-games agents".
Multimedia & Intelligent Systems Research Laboratory
University of Wolverhampton, United Kingdom

[16] Eiben, A.E., Smith, J.E. (2003)
"Evolutionary Computing Tutorial Part 1: What is an Evolutionary Algorithm?".
http://www.evonet.polytechnique.fr/CIRCUS2/
valise/bcraenen/intro_tut.pdf

[17] Eiben, A.E., Smith, J.E. (2003)
"Lecture Slides - What is an Evolutionary Algorithm?".
http://www.evonet.polytechnique.fr/CIRCUS2/valise/bcraenen/EC-tutorial-whatis.ppt

[18] FIPA (2001)
"The foundation for intelligent physical agents".
http://www.fipa.org

[19] Gosavi, A. (2003)
"Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning".
Kluwer Academic Publishers
ISBN: 1402074549

[20] Hahn, A.D. (2003)
"The Fortress of infinitude - A random repository of interests and information".
http://www.randomterrain.com/gamedesign/randomness.html

[21] Herbert, F. (1965)
"Dune".
Philadelphia, PA: Chilton Books
ISBN: 0441172717

[22] Jansen R.P.S. et al. (1998)
"The bottleneck: mitochondrial imperatives in oogenesis and ovarian follicular fate".
Molecular and Cellular Endocrinology 145, pp. 81-88

[23] Johnson-Laird, P.N. (1997)
"De computer en de menselijke geest - Een inleiding in de cognitieweten-schap".
FontanaPress/HarperCollins
ISBN: 90-274-4165-0

[24] Kernighan, B., Ritchie, D.M. (1989)
"C Handboek".
Prentice Hall - Academic Service
ISBN: 90-6233-488-1

[25] Laird, J.E. (2000)
"It Knows What You're Going to Do: Adding Anticipation to a Quakebot".
Proceedings of the AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, March 2000 (AAAI technical report #SS-00-02).

[26] Laird, J.E., Van Lent, M. (2000)
"Interactive Computer Games: Human-Level AI's Killer Application".
Proceedings of the AAAI National Conference on Artificial Intelligence, August 2000.

[27] Mitchell, T.M. (1997)
"Machine Learning"., pp. 81-127, 368-390

McGraw-Hill Science/Engineering/Math
ISBN: 0070428077

[28] Nilsson, N.J. (1982)
"Principles of Artificial Intelligence".
Springer-Verlag, 2nd edition, Berlin

[29] Pietro, A. D., While, L., Barone, L. (2002)
"Learning in RoboCup keepaway using evolutionary algorithms".
In W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, GECCO 2002.

[30] Pfeifer, R., Scheier, C. (2002)
"Understanding Intelligence".
The MIT Press, Cambridge, Massachusetts, London, England
ISBN: 0-262-66125-X

[31] *PlanetQuake* (2003)
"The epicenter of everything Quake".
http://www.planetquake.com

[32] Postma, E.O. (1994)
"SCAN: A Neural Model of Covert Attention".
Universiteit Maastricht, The Netherlands
ISBN: 90-9007320-5

[33] Pyeatt, L.D. et al. (1998)
"Learning to Race: Experiments with a Simulated Race Car".
Colorado State University

[34] Rabin, S. (2002)
"AI Game Programming Wisdom".
First Edition, Charles River Media
ISBN: 1-58450-077-8

[35] Rijswijck, J., van (2003)
"Learning Goals in Sports Games".
Game Developers Conference, San Jose, 2003
Department of Computing Science, University of Alberta

[36] Robbins, T (1980)
"Still Life With Woodpecker".
Bantom, New York, pp. 19

[37] Robert, G., Portier, P., Guillot, A. (2002)
"Classifier systems an 'animat' architectures for action selection in MMORPG".
AnimatLab, Laboratoire d'Informatique de Paris, France

[38] Spronck, P.H.M., Sprinkhuizen-Kuyper, I.G., Postma, E.O. (2003)
"Online Adaptation of Computer Game Opponent AI".
Universiteit Maastricht, The Netherlands

[39] Spronck, P.H.M., Sprinkhuizen-Kuyper, I.G., Postma, E.O. (2002)
"Improving Opponent Intelligence by Machine Learning".
Universiteit Maastricht, The Netherlands

[40] Sterren, W., van der (2002)
"Squad Tactics: Team AI and Emergent Maneuvers".
AI Game Programming Wisdom (ed. Rabin, S.), pp. 233-246
Charles River Media

[41] Stroustrup, B. (1997)
"The C++ Programming Language".
Third Edition, Addison-Wesley
ISBN: 0-201-88954-4

[42] Taylor, C. (2002)
"President of Gas Powered Games, shares his opinions with us about the future of PC gaming".
GameSpy.com, Dec. 7, 2002
http://www.gamespy.com/interviews/december02/christaylor/index.shtml

[43] Tozour, P. (2002)
"The Evolution of Game AI".
AI Game Programming Wisdom (ed. Rabin, S.), pp. 3-15
Charles River Media

[44] Tozour, P. (2002)
"The Perils of AI Scripting".
AI Game Programming Wisdom (ed. Rabin, S.), pp. 541-547
Charles River Media

[45] Waveren, J.P.M. van, (2001)
"The Quake III Arena Bot".
Master's thesis, revision 1
University of Technology Delft, Faculty ITS

[46] Waveren, J.P.M. van, and Rothkrantz, L.J.M. (2001)
"Artificial Player for Quake III Arena".
2nd International Conference on Intelligent Games and Simulation GAME-ON 2001 (eds. Quasim Mehdi, Norman Gough and David Al-Dabass).
SCS Europe Bvba, pp. 48-55

[47] Weizenbaum, J. (1966)
"ELIZA – A computer program for the study of natural language communication between man and machine".
Communications of the ACM 9(1), pp. 36-45

[48] Winands, M.H.M. (2000)
"Analysis and Implementation of Lines of Action".
Master's Thesis CS 00-03
Universiteit Maastricht, The Netherlands

[49] Woodcock, S. (2000)
"Game AI: The State of the Industry".
Gamasutra
http://www.gamasutra.com/features/20001101/woodcock_01.htm

[50] Wooldridge, M. (2002)
"An Introduction to Multiagent Systems", pp. 214-221.
John Wiley & Sons, Chichester, England
ISBN: 0 47149691X

[51] Wyrobek, A.J. et al. (1996)
"Mechanisms and Targets Involved in Maternal and Paternal Age Effect on Numerical Aneuploidy".
Biology and Biotechnology Research Program, L-452, Lawrence Livermore National Laboratory
Environmental and Molecular Mutagenesis 28, pp. 254-264

Appendix A

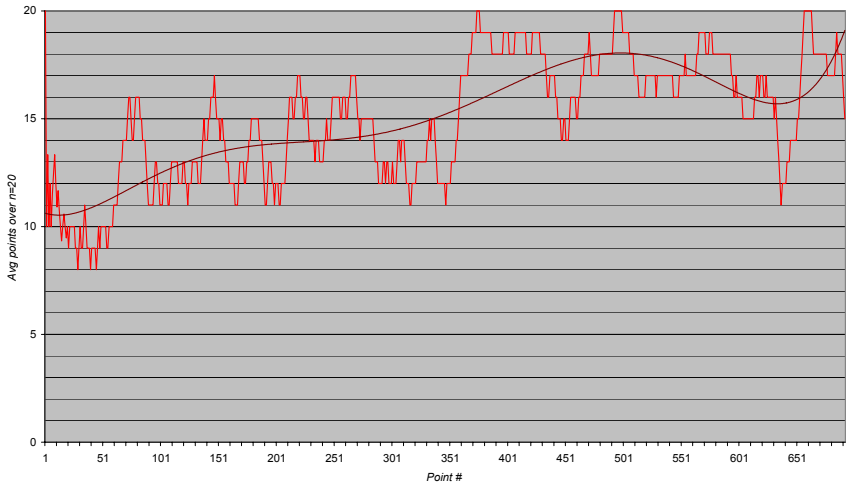
Experimental results TEAM vs Static Opponent

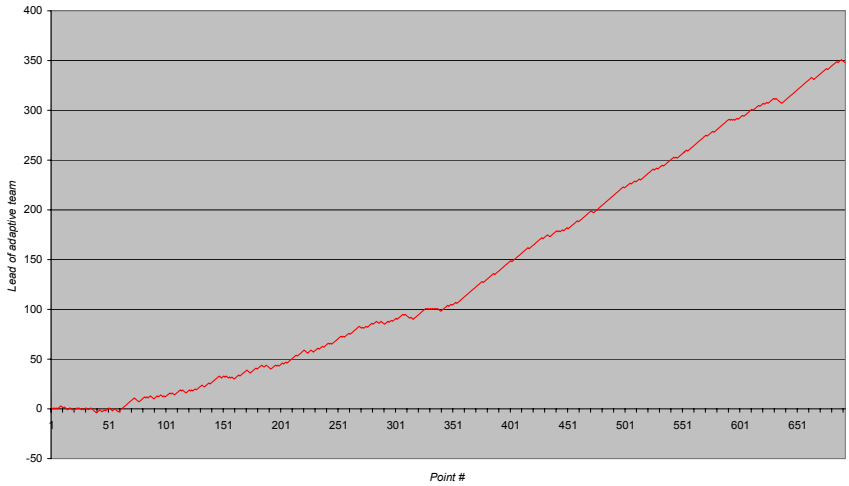
Experiment #	Absolute turning point	Relative turning point
0	72	62
1	53	8
2	150	144
3	70	52
4	137	126
5	173	12
Avg	109.17	67.33
StDev	50.17	56.86
StError	15.13	17.14
Median	104.5	57
Minimum	53	8
Maximum	173	144
Avg Low	94	50
Avg High	124	84

Figure A.1: Summary of experiment results.

A.1 Experimental run #0 (Long run)

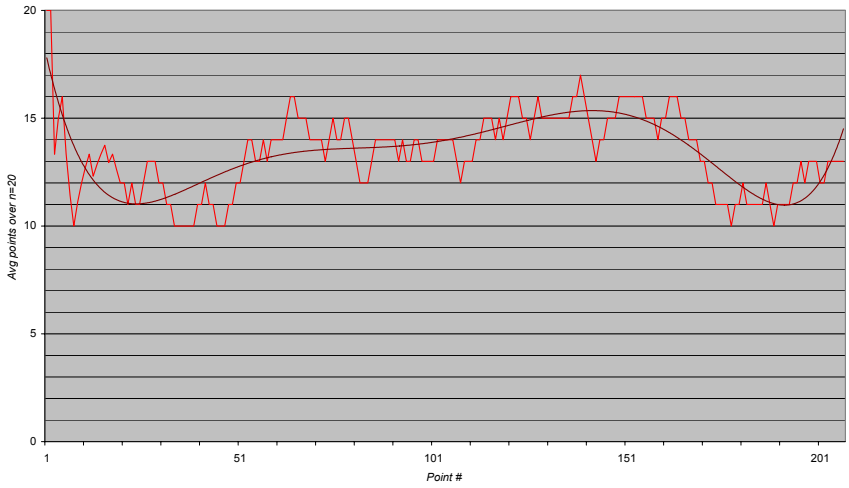
Score total	692
Score adaptive team	520
Score non-adaptive team	172
Absolute turning point	72
Relative turning point	62

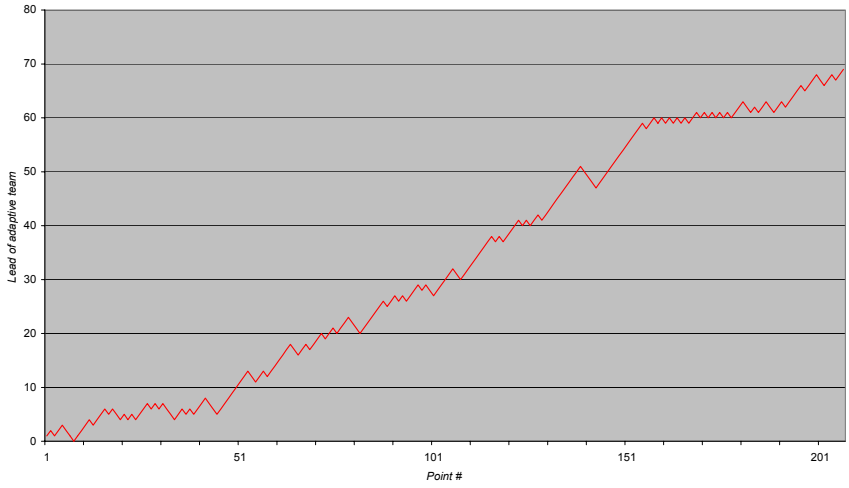




A.2 Experimental run #1

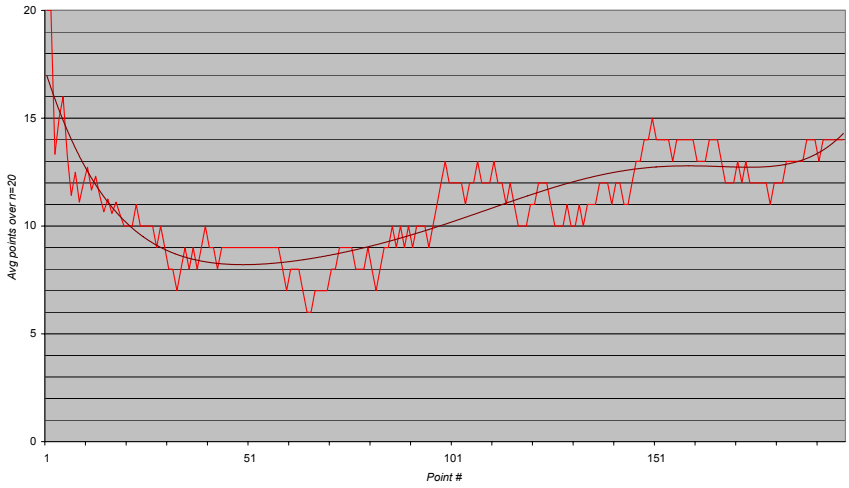
Score total	207
Score adaptive team	138
Score non-adaptive team	69
Absolute turning point	63
Relative turning point	8

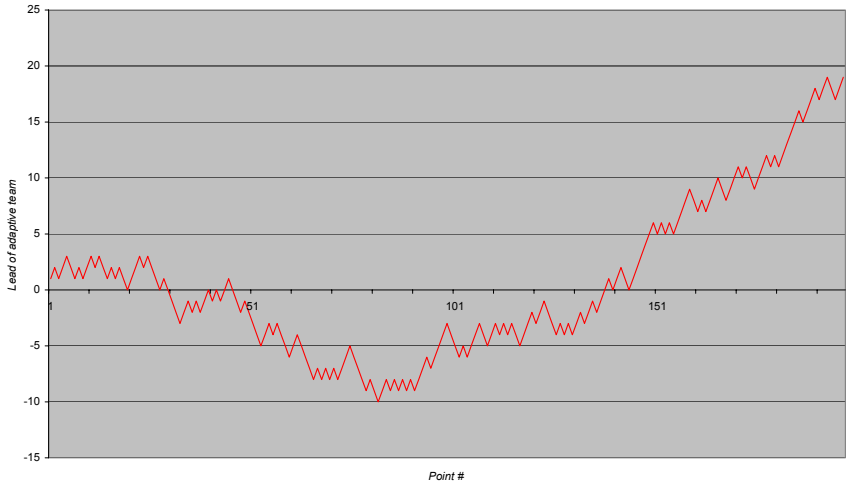




A.3 Experimental run #2

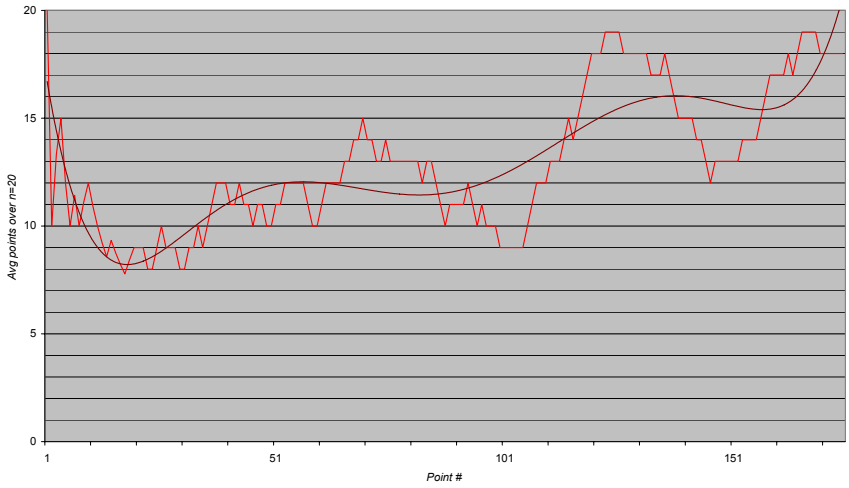
Score total	197
Score adaptive team	108
Score non-adaptive team	89
Absolute turning point	150
Relative turning point	144

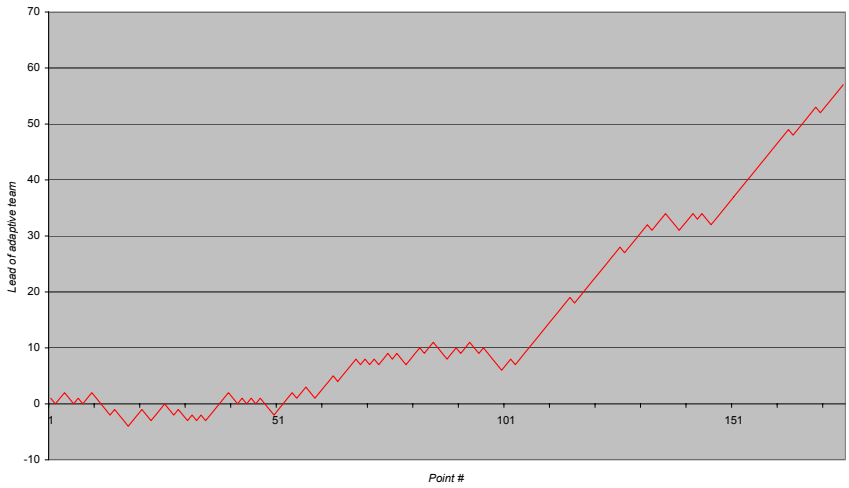




A.4 Experimental run #3

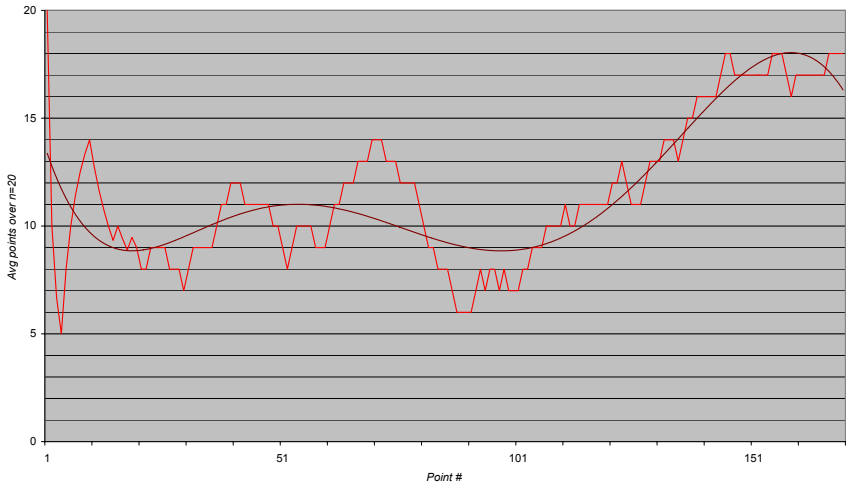
Score total	175
Score adaptive team	116
Score non-adaptive team	59
Absolute turning point	70
Relative turning point	52

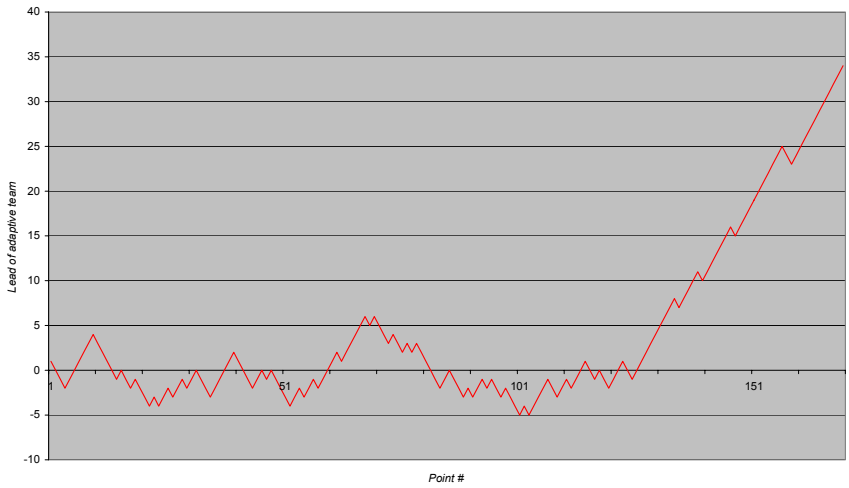




A.5 Experimental run #4

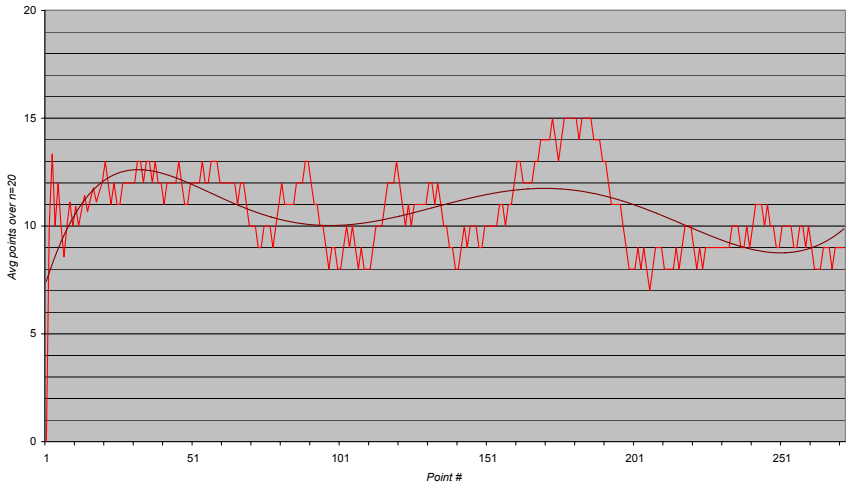
Score total	170
Score adaptive team	102
Score non-adaptive team	68
Absolute turning point	137
Relative turning point	126

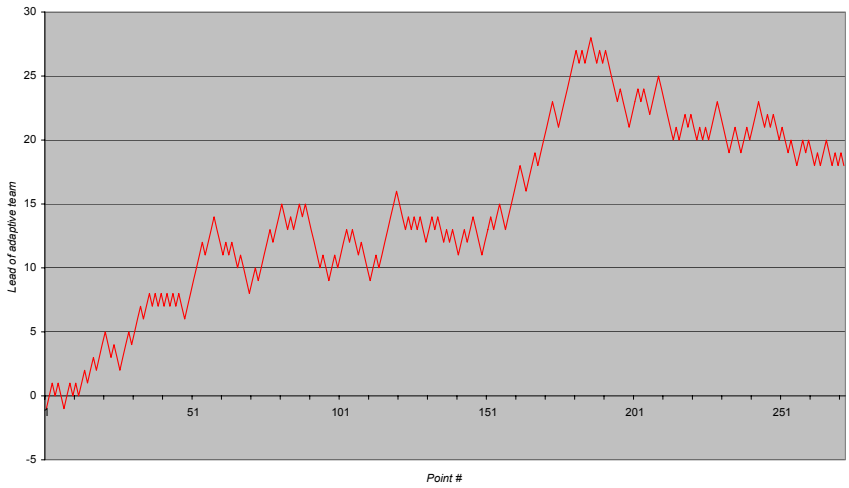




A.6 Experimental run #5

Score total	272
Score adaptive team	145
Score non-adaptive team	127
Absolute turning point	173
Relative turning point	12





Appendix B

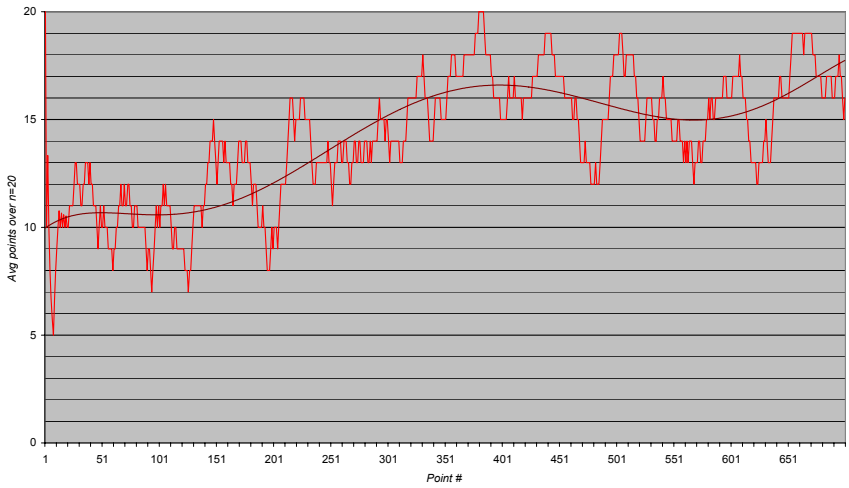
Experimental results TEAM vs Original Quake III Opponent

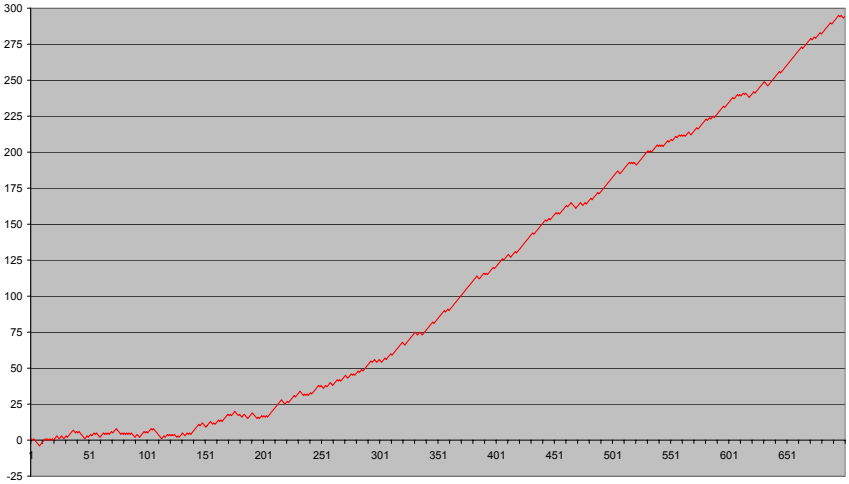
Experiment #	Absolute turning point	Relative turning point
1	148	20
2	263	158
3	106	70
4	38	36
5	99	50
6	42	24
7	58	114
8	107	48
9	205	132
10	92	32
11	61	56
12	127	144
13	136	50
14	91	82
15	53	54
Avg	108.40	71.33
StDev	61.99	44.78
StError	18.69	13.50
Median	99	50
Minimum	38	20
Maximum	263	158
Avg Low	90	58
Avg High	127	85

Figure B.1: Summary of experiment results.

B.1 Experimental run #1

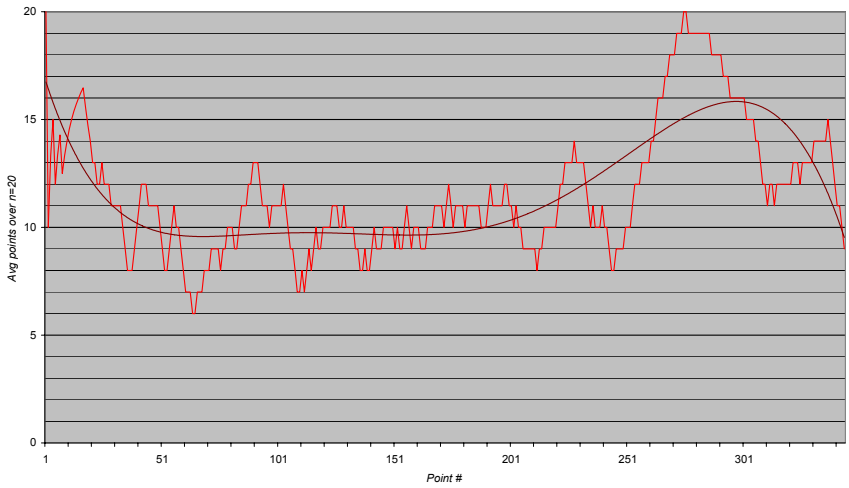
Score total	700
Score adaptive team	497
Score non-adaptive team	203
Absolute turning point	148
Relative turning point	20

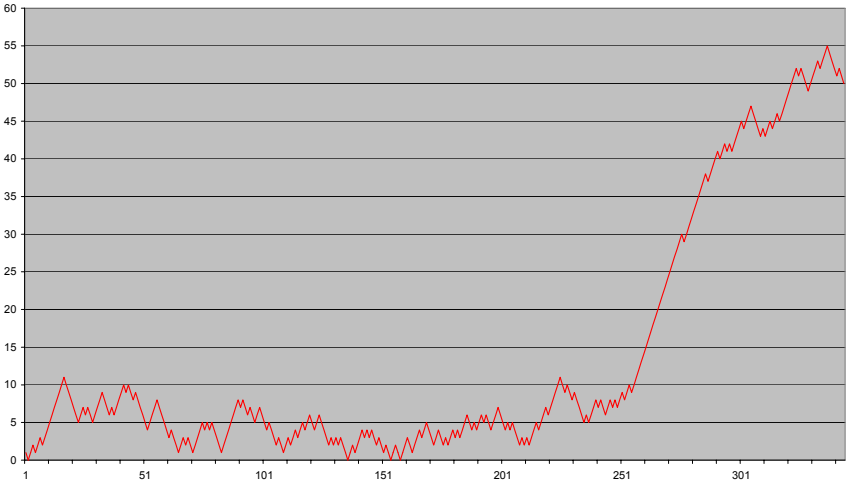




B.2 Experimental run #2

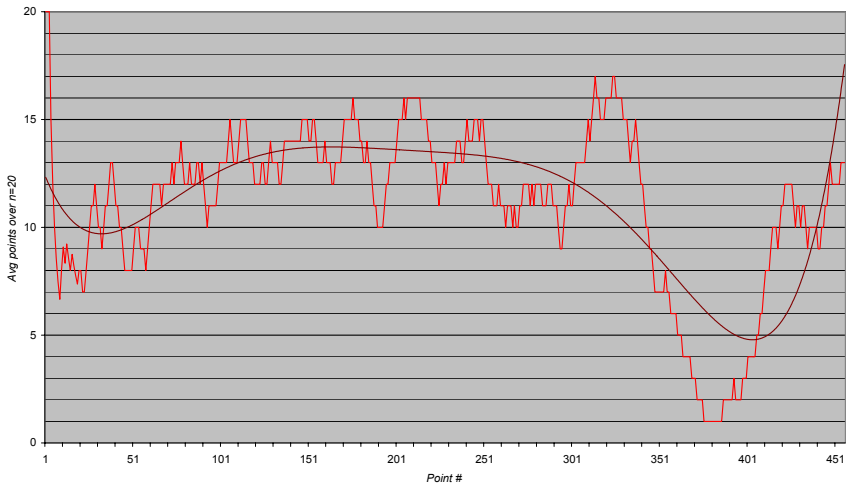
Score total	344
Score adaptive team	197
Score non-adaptive team	147
Absolute turning point	263
Relative turning point	158

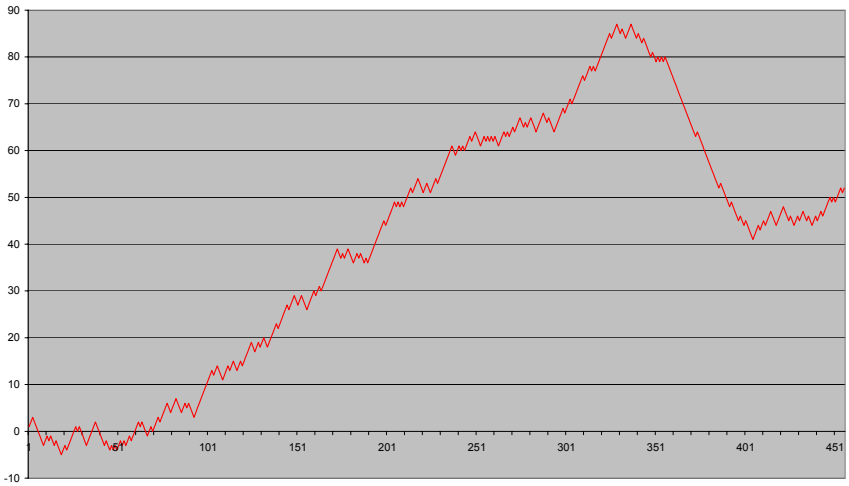




B.3 Experimental run #3

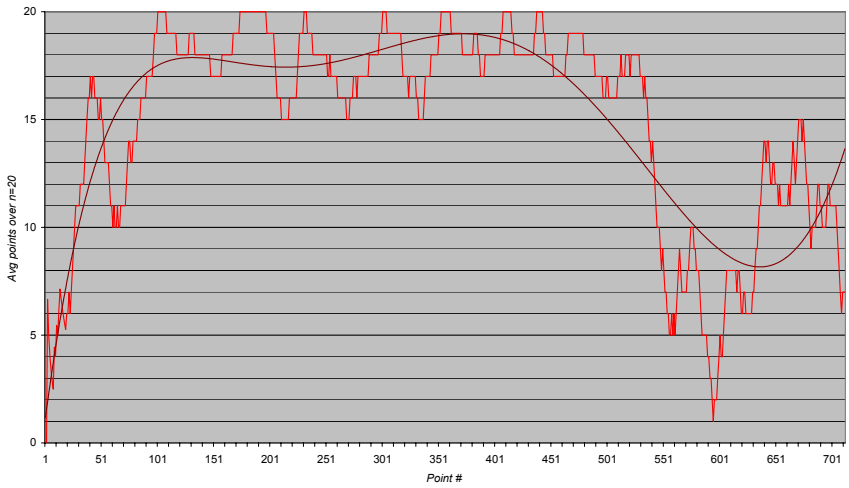
Score total	456
Score adaptive team	254
Score non-adaptive team	202
Absolute turning point	106
Relative turning point	70

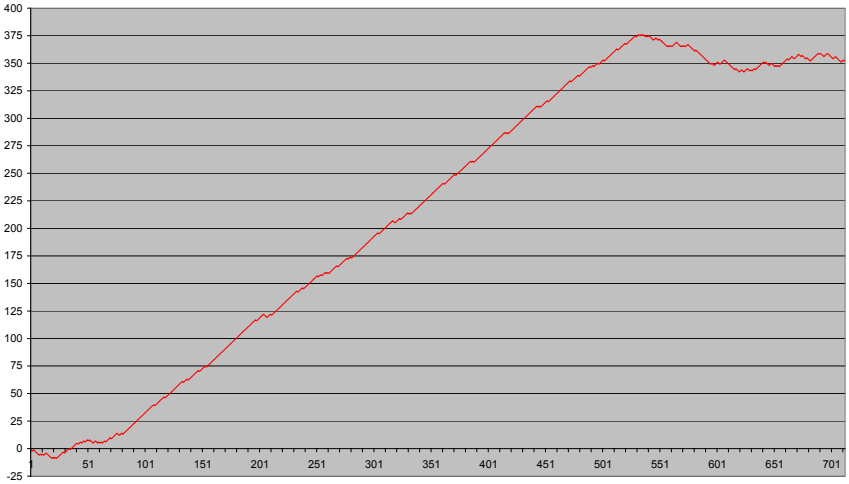




B.4 Experimental run #4

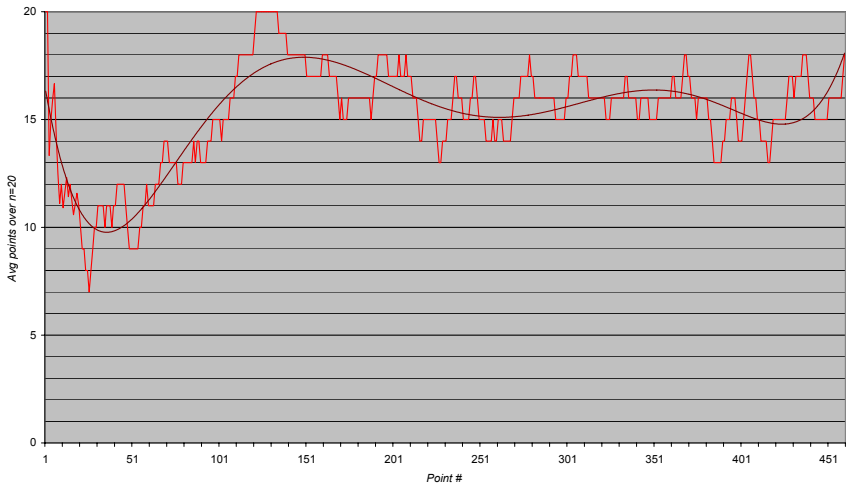
Score total	712
Score adaptive team	532
Score non-adaptive team	180
Absolute turning point	38
Relative turning point	36

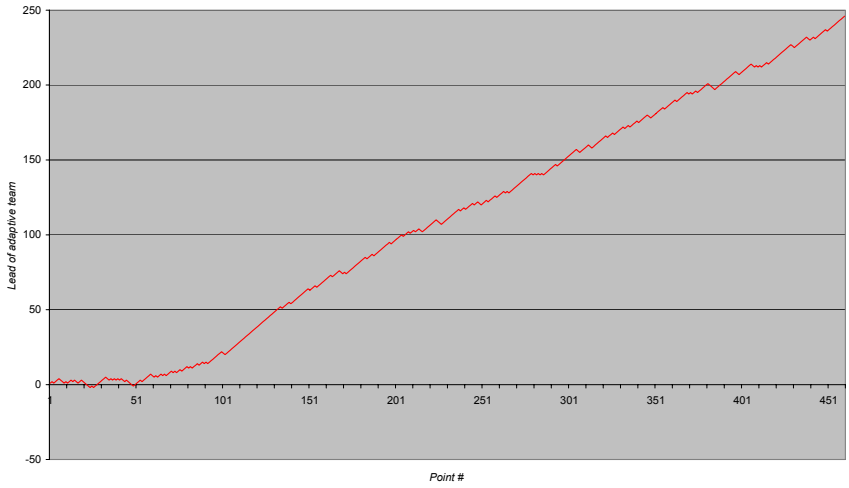




B.5 Experimental run #5

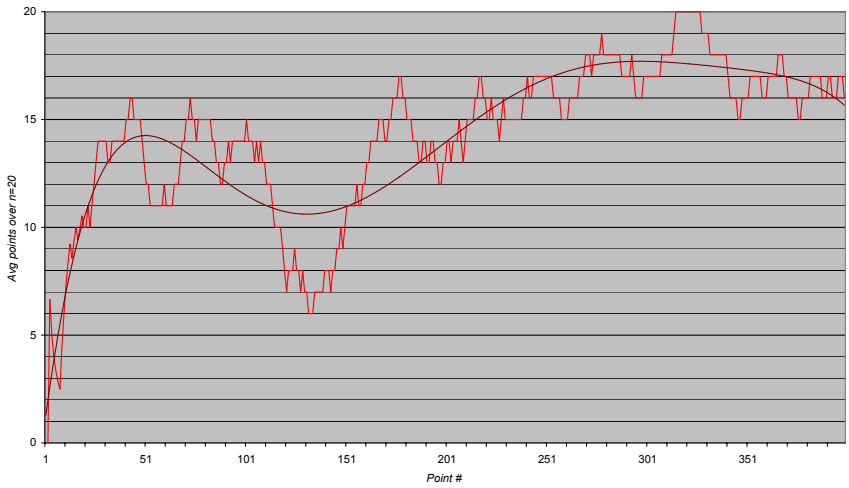
Score total	460
Score adaptive team	353
Score non-adaptive team	107
Absolute turning point	99
Relative turning point	50

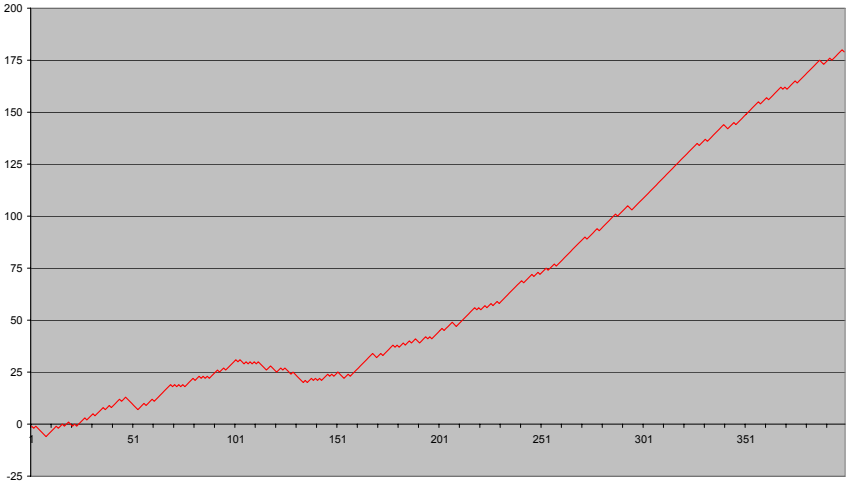




B.6 Experimental run #6

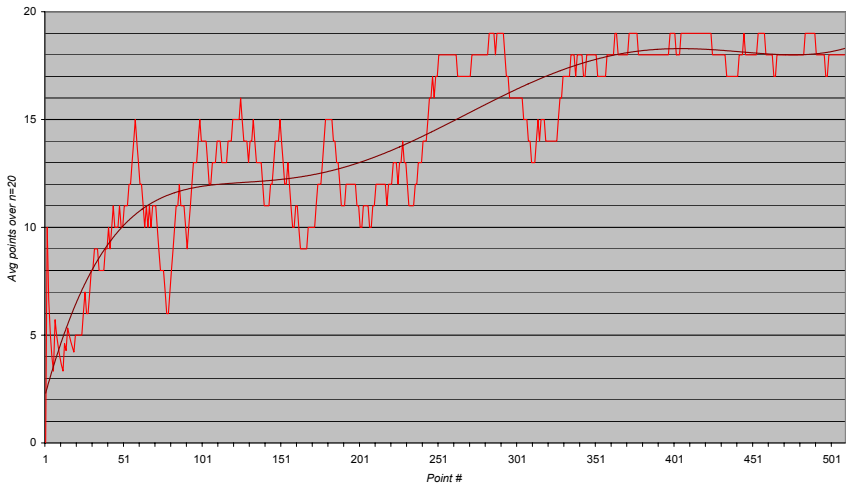
Score total	399
Score adaptive team	289
Score non-adaptive team	110
Absolute turning point	42
Relative turning point	24

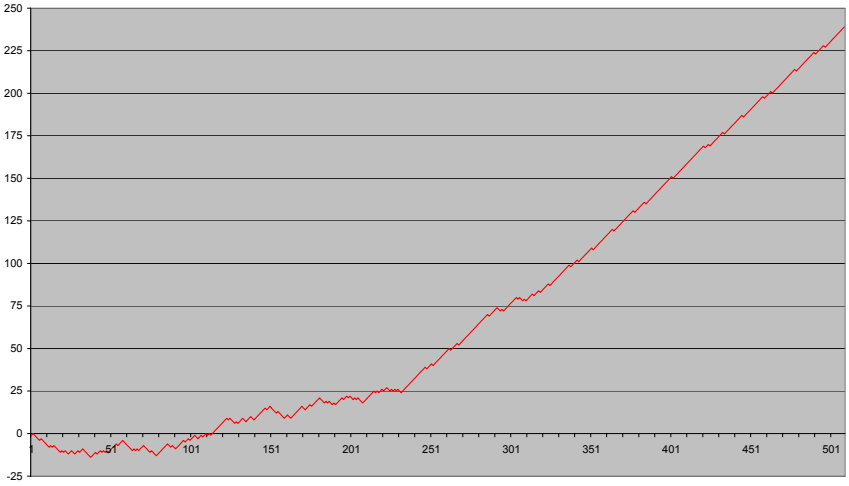




B.7 Experimental run #7

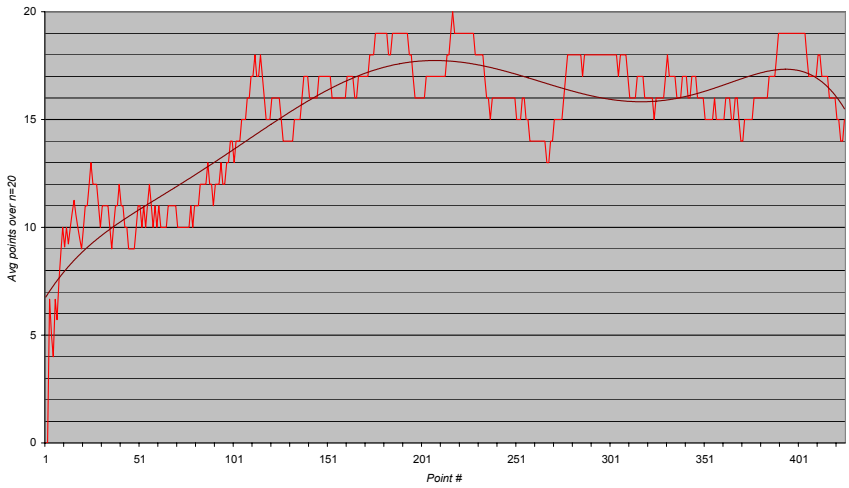
Score total	509
Score adaptive team	374
Score non-adaptive team	135
Absolute turning point	58
Relative turning point	114

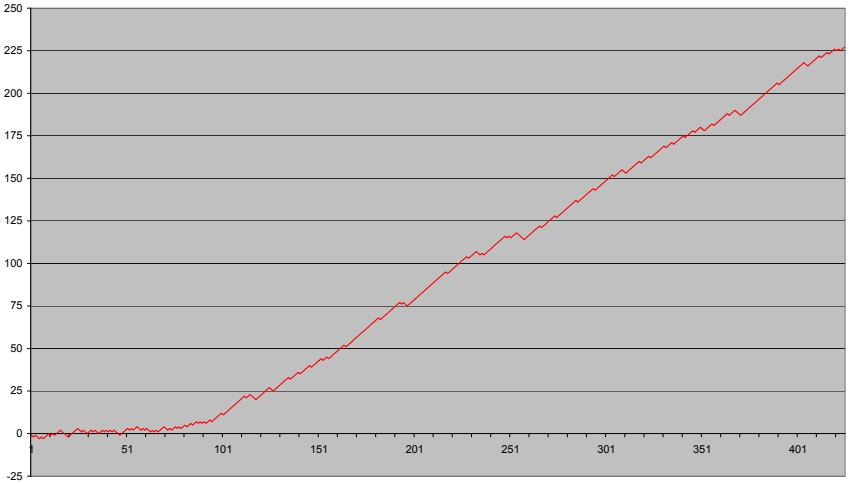




B.8 Experimental run #8

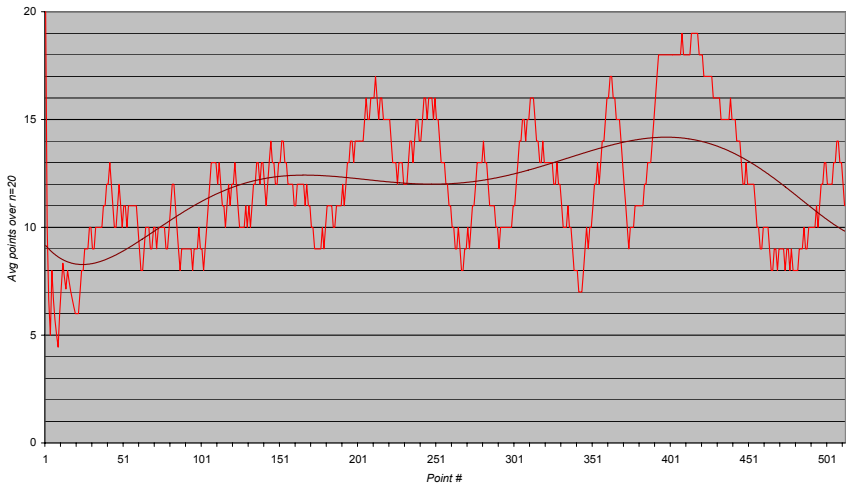
Score total	425
Score adaptive team	326
Score non-adaptive team	99
Absolute turning point	107
Relative turning point	48

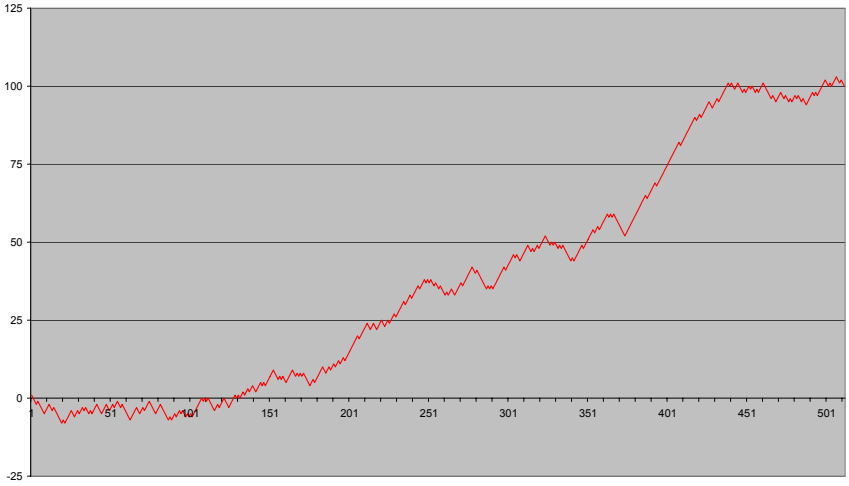




B.9 Experimental run #9

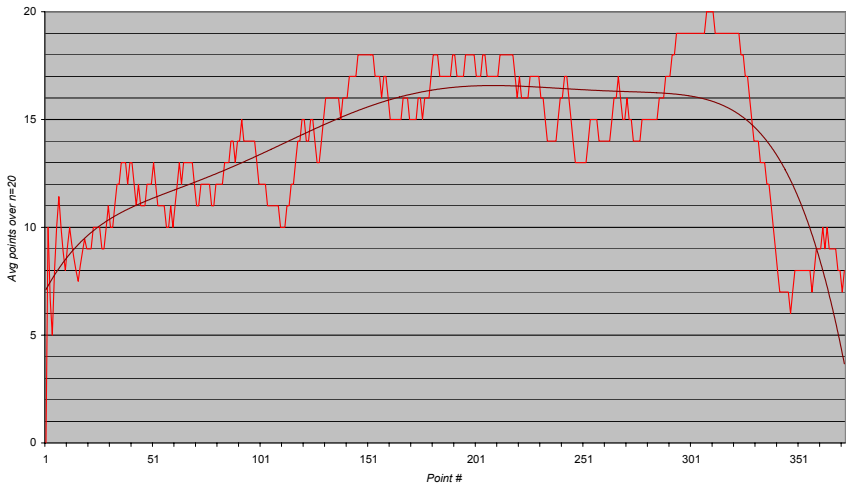
Score total	512
Score adaptive team	306
Score non-adaptive team	205
Absolute turning point	205
Relative turning point	132

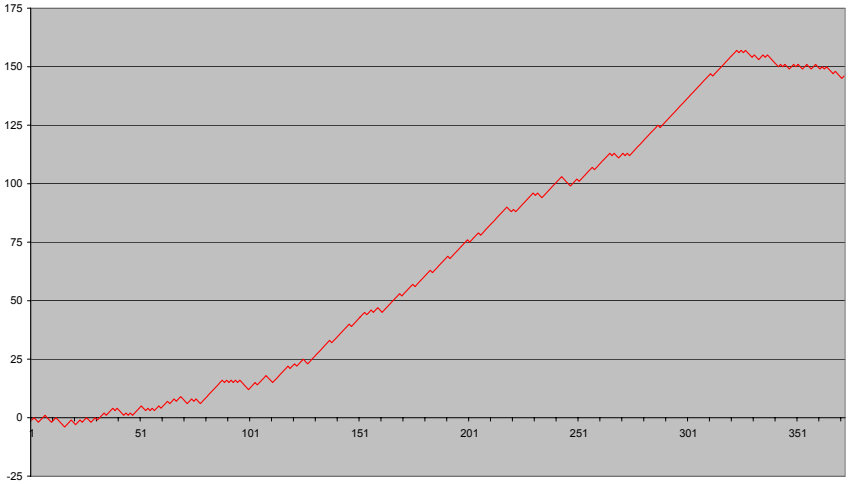




B.10 Experimental run #10

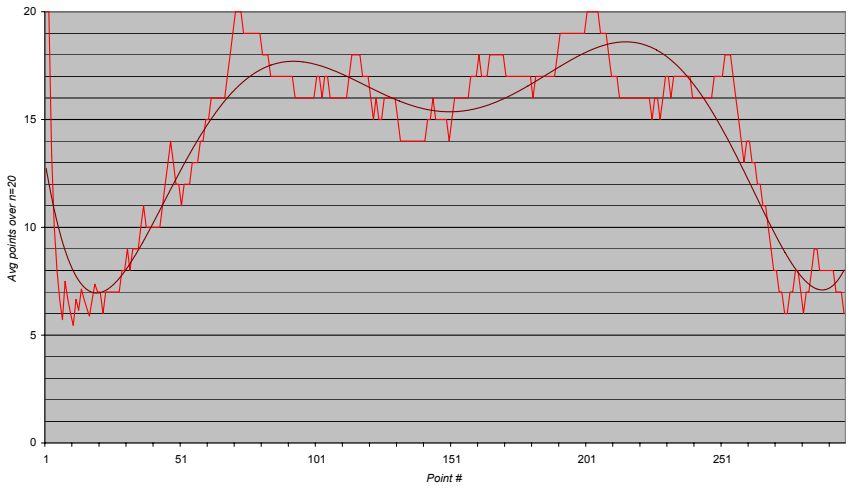
Score total	372
Score adaptive team	259
Score non-adaptive team	113
Absolute turning point	92
Relative turning point	32

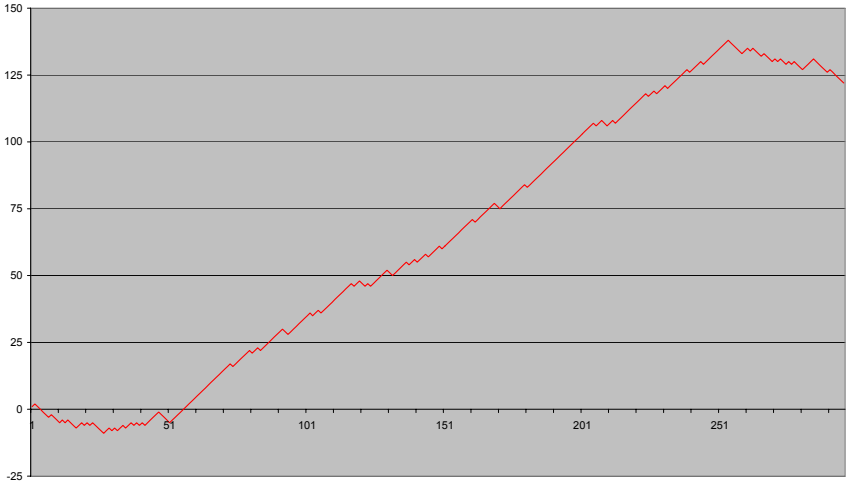




B.11 Experimental run #11

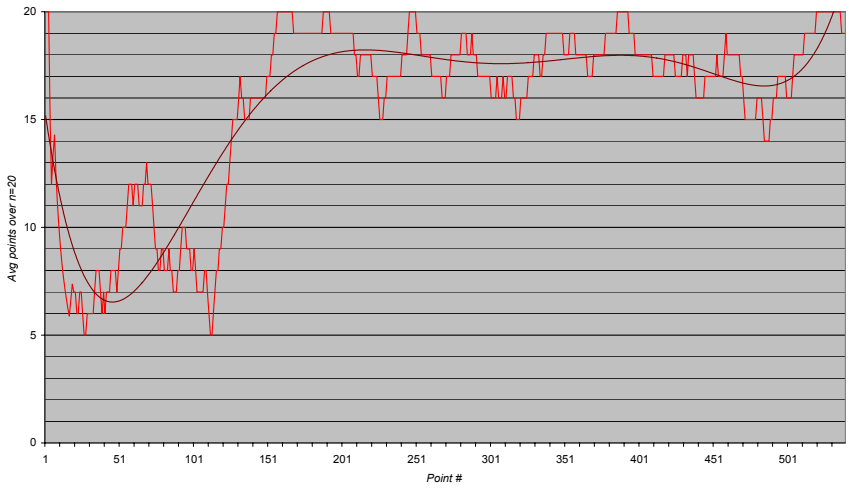
Score total	296
Score adaptive team	209
Score non-adaptive team	87
Absolute turning point	61
Relative turning point	56

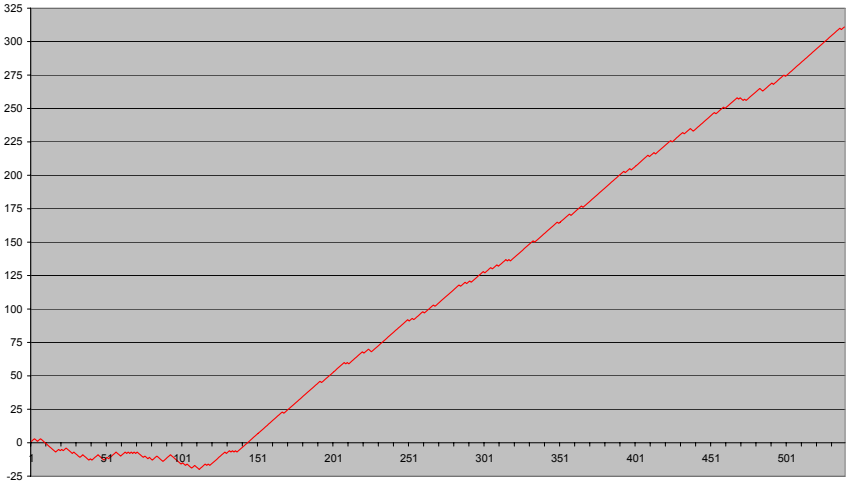




B.12 Experimental run #12

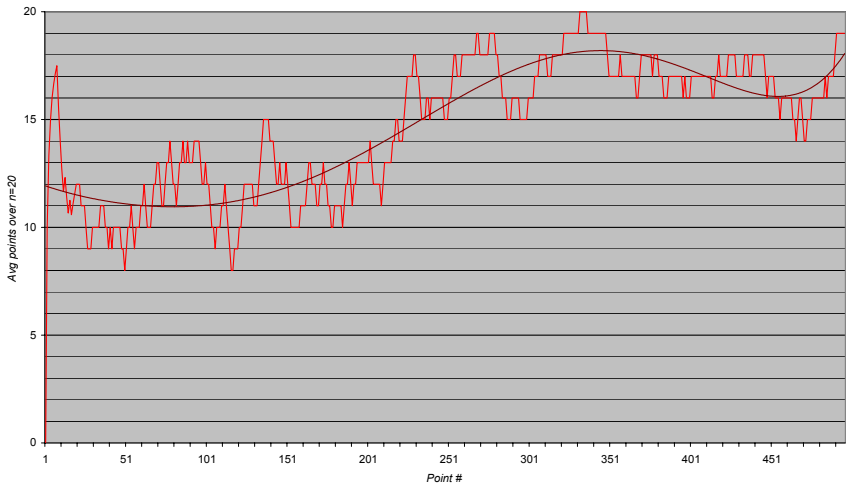
Score total	539
Score adaptive team	425
Score non-adaptive team	114
Absolute turning point	127
Relative turning point	144

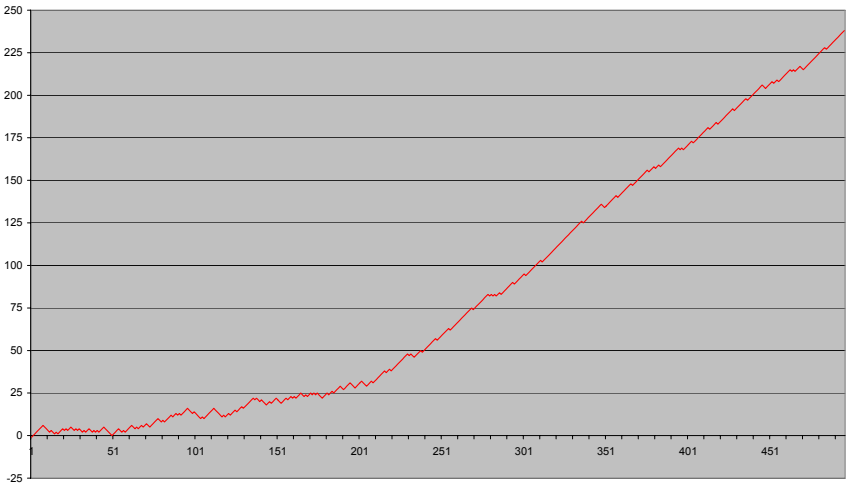




B.13 Experimental run #13

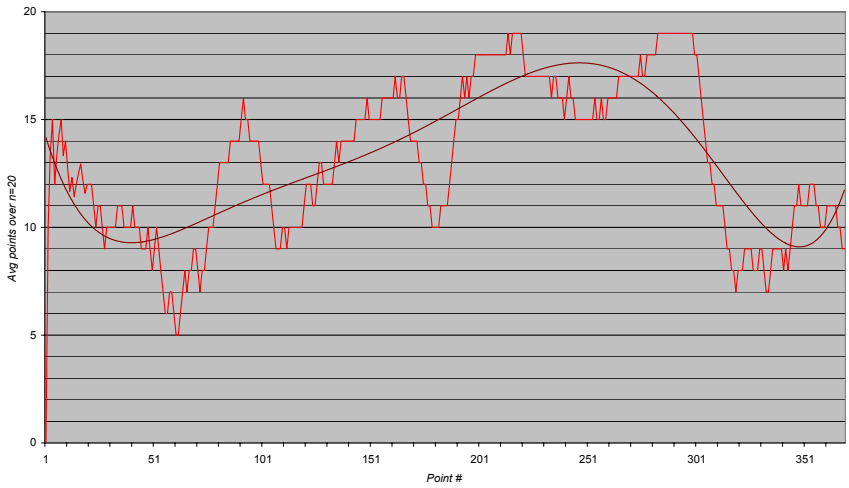
Score total	496
Score adaptive team	367
Score non-adaptive team	129
Absolute turning point	136
Relative turning point	50

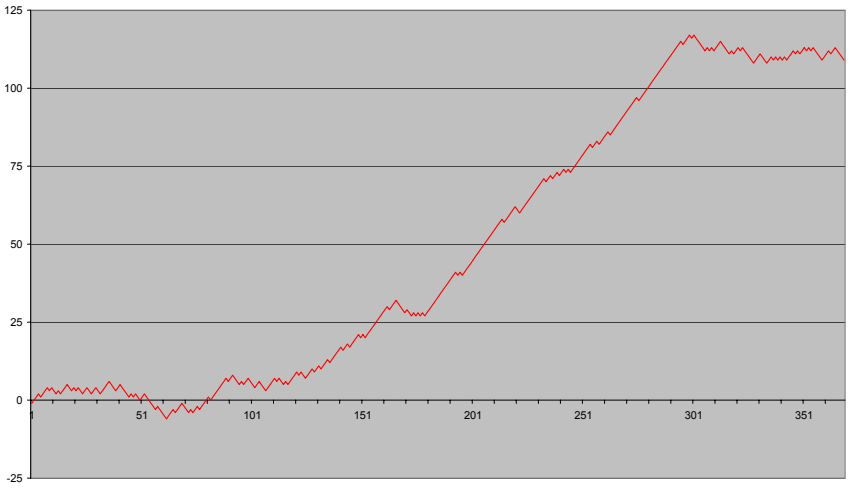




B.14 Experimental run #14

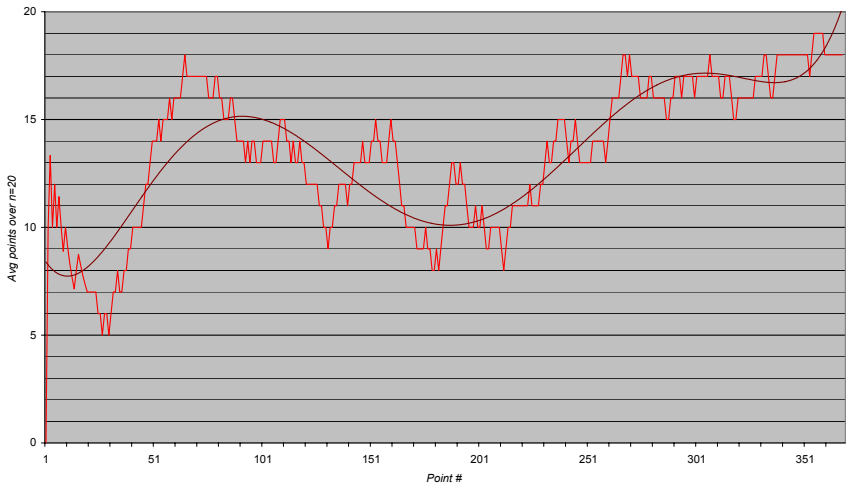
Score total	369
Score adaptive team	239
Score non-adaptive team	130
Absolute turning point	91
Relative turning point	82

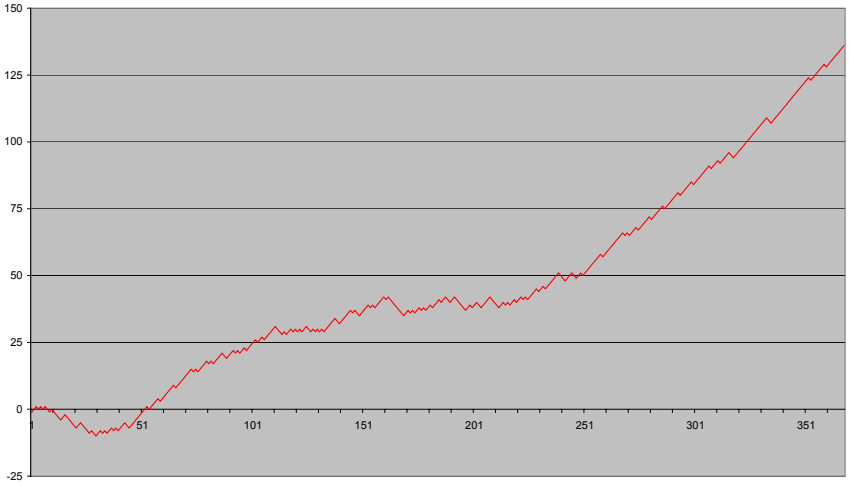




B.15 Experimental run #15

Score total	368
Score adaptive team	252
Score non-adaptive team	116
Absolute turning point	53
Relative turning point	54





Appendix C

Forgetting tactics

This appendix presents an example of TEAM 'forgetting' tactics. Data is used from state 'Both flags at their base' of the experimental run which is displayed in appendix B.4.

Observe figure C.1. Initially, the population consists of genomes with high fitness values. Due to unlucky circumstances the fitness value of the genomes is recalculated to lower values, and replaced by genomes which seem superior. Subsequently, as can be observed, the fitness value of the new genomes is also recalculated to lower values. The effect of this process is that in a short period of time the population consisting of genomes which are superior (but do not seem to be) is replaced by a population of genomes which seem superior (but are not).

	Slot1		Slot2		Slot3		Slot4		Slot5	
Point	**Genome**	**Fitness**	**Genome**	**Fitness**	**Genome**	**Fitness**	**Genome**	**Fitness**	**Genome**	**Fitness**
532	*1341*	*1.27*	*1317*	*1.36*	*1329*	*1.26*	*1330*	*1.13*	*1344*	*1.40*
		0.84		1.32		0.91	1346	1.25		1.33
		0.79		1.07		0.95		0.83		1.05
				1.37	1361	1.11		0.71	1348	1.52
							1359	1.66		1.16
								1.39		0.81
								0.99	1352	0.83
									1360	1.08
537	*1356*	*0.99*		*0.88*		*0.83*		*0.96*		*0.93*
		0.82	1371	1.16				0.99		0.56
		1.12		1.05					1368	0.66
		0.96		0.99					1369	1.41
										1.29
										0.86
										0.85
									1374	0.93
									1376	1.35
541		*1.16*	*1379*	*1.30*	*1370*	*1.02*	*1380*	*1.17*		*1.21*
		0.78		1.03	1382	1.20		0.84		1.02
		0.65		0.61		1.17				0.88
	1393	0.85		0.38		1.05			1385	1.14
		0.93	1396	0.45		0.72				0.86
		0.59	1397	1.01		0.56				0.65
	1400	0.93		0.66	1405	0.90				
		0.62		0.80		0.70				
		0.48		0.88		0.46				
	1404	0.96		0.89						
		0.92								
		0.68								
546		*0.66*		*0.73*	*1411*	*0.57*		*0.65*	*1406*	*0.89*

Figure C.1: Example of the occurance of a degrading quality of the population.

Summary

In the short history of computer science, 'ancient' commercial computer games, such as Pacman, demonstrated that an entertaining commercial computer game requires a rudimentary form of artificial intelligence. Throughout the years, commercial computer games became increasingly realistic with regard to the visual and auditory presentation. However, artificial intelligence in commercial computer games has not yet obtained a high degree of realism.

We observed that current commercial computer-game opponents are endowed with inferior team-oriented behaviour. The observation has led us to the following problem statement: is it possible to improve the performance of opponents in state-of-the-art commercial computer games, with regard to their team-oriented behaviour?

One reason for the inferiority of team-oriented behaviour in commercial computer games is that it lacks adaptive team AI. We aim at creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment.

An approach to deal with this aim is to create a mechanism which imposes adaptive team AI on commercial computer-game opponents. The following research question guided our research: is it possible to create a mechanism that imposes adaptive team AI on commercial computer-game opponents and achieves a qualitatively acceptable performance?

In our attempt to answer the research question, we had three objectives: (1) designing a mechanism that imposes adaptive team AI on opponents in commercial computer games, (2) implementing the design in a test environment, and (3) obtaining a qualitatively acceptable performance of the adaptive mechanism, i.e., performance that is computationally fast, robust, efficient and effective [38].

In accordance with the first research objective, we designed a team-oriented adaptive mechanism and named it the *Tactics Evolutionary Adaptability Mechanism* (TEAM).

In accordance with the second research objective, we successfully implemented the design of TEAM in Quake III, a state-of-the-art commercial computer-game.

In accordance with the third research objective, we evaluated the requirements for qualitatively acceptable performance [38], and drew the conclusion that TEAM is computationally fast, robust, efficient and effective. Thereupon,

drew the conclusion that TEAM obtained a qualitatively acceptable performance.

By achieving all our research objectives, we drew a final conclusion by answering the research question with an unequivocal yes, it is indeed possible to create a mechanism that imposes adaptive team AI on commercial computer-game opponents and achieves a qualitatively acceptable performance.

Our approach to answering the problem statement was aimed at creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment. Taking the answer to the research question above into consideration, the answer to the problem statement must be that it is indeed possible to improve the performance of opponents in state-of-the-art commercial computer games, with regard to their team-oriented behaviour.

TEAM is capable of unsupervised and intelligently adapting to the environment. TEAM adapted to the environment in such a way that it evolved towards dominant tactics. These dominant tactics, which resulted in 'dangerous' but successful behaviour, exceeded the vision of the designers of Quake III, which was focussed at always behaving moderately. Therefore, we have fulfilled our aim of creating adaptive team AI capable of exceeding the limitations of its designer's vision by unsupervised and intelligent adaptation to the environment.

Adaptive team AI in commercial computer games does not exist. Therefore, our results provided a significant contribution to the study and application of artificial intelligence techniques in general, and machine-learning techniques in commercial computer games in particular.