Dynamic Formations in Real-Time Strategy Games

Marcel van der Heijden, Sander Bakkes, and Pieter Spronck

Abstract— **Current approaches to organising units in strategic video games are typically implemented via static formations. Static formations are not capable of adapting effectively to opponent tactics. In this paper we discuss an approach to organising units by learning the effectiveness of a formation in actual play, and directly applying learned formations according to the classification of the opponent player. This approach to establish so-called dynamic formations, is tested in the ORTS game environment. From our results, we may conclude that the approach to established dynamic formations can be successfully applied in actual video-game environments.**

I. Introduction

The effectiveness of AI in video games depends heavily on how well game characters are able to cooperate and react to the opponent player. In many video games, this behaviour is implemented via so-called formations. Today, formations are expected for any type of cohesive group behaviour. From squad-based first-person shooters to sport simulations to real-time strategy game, anytime that a group is moving or working together it is expected to do so in an orderly, intelligent fashion [1]. Approaches exist to establish effective formations. However, these approaches are typically built upon static techniques. This renders the so established formations unable to adapt to changing circumstances. To allow formations to be effective in changing (and unforeseen) circumstances, in this paper we describe a novel approach to establish formations *dynamically*, based on previous experiences with the game.

The outline of this paper is as follows. In Section II we give background information. In Section III we discuss a general framework to establish dynamic formations. Section IV describes how to incorporate a learning algorithm to select which dynamic formation to employ best. In Section V we discuss how to establish and utilise models of the opponent player for the purpose of effectively applying learned behaviour. In Section VI we discuss experiments that test our approach to dynamic formations in actual play, together with the experimental results. Section VII provides conclusions and describes future work.

II. Background

In this section we give background information with regard to (A) formations, (B) learning in video games, and (C) the ORTS game environment for AI research.

Marcel van der Heijden (email: m_v_d_heijden@hotmail.com) received his M.Sc. degree from Maastricht University in 2008. Sander Bakkes and Pieter Spronck are affiliated to the Tilburg centre for Creative Computing (TiCC), Tilburg University, The Netherlands (phone: +31 13 466 8118; fax: +31 13 466 2892 ; email: {s.bakkes, p.spronck}@uvt.nl).

A. Formations

A formation is defined as an arrangement or disposition of units [2]. Formations are typically applied for a tactical purpose, and have already been found in tribal societies such as the Maori [3]. Commonly seen formations, such as a shield wall, a phalanx or a wedge, have historical significance and are still used in modern military operations [1].

Throughout the years, video games grew to encompass tactical realism to include formations, such as in the popular game AGE OF EMPIRES III. To establish formations in video games, there are two approaches that are typically applied [4]. The first approach deals with fixed formations, in which each unit is assigned a fixed slot in a predefined formation. Formations established via this approach are static, which implies that they are generally unsuitable for unforeseen circumstances. The second approach deals with emergent formations, in which formations emerge autonomously from the interaction of units. The formations established via this approach, though not static, are difficult to control and to predict in behaviour.

These typical approaches are not suitable to achieve the goal of establishing automatically formations that may effectively be applied in strategic video games, such as RTS games. To allow formations to be effective even in changing circumstances, in the sections that follow we discuss a novel approach to establish formations *dynamically*, based on previous experiences with the game.

B. Learning in Video Games

As modern video games present a complex and realistic environment, one would expect characters controlled by game AI in such an environment to behave realistically ('human-like') too. One important feature of human-like behaviour of game AI is the ability to adapt to changing circumstances. Game AI with this ability is called 'adaptive game AI', and is typically implemented via machine-learning techniques. Adaptive game AI may be used to significantly improve the quality of game AI by learning effective behaviour while the game is in progress. Adaptive game AI has been successfully applied to simple video games [5], [6], and to complex video games [7].

We observe, however, that learning effective behaviour while the game is in progress (i.e., 'online'), typically requires an inefficiently large number of learning trials. It is not uncommon that a game has finished before effective behaviour could be established, or that game characters in a game do not live long enough to benefit from learning. As a result, it is difficult for players to perceive that game AI in fact is learning. This renders the benefits of online learning in video games subjective and unclear [8]. In our approach to

Fig. 1. Screenshot of the ORTS game environment.

dynamic formations, we therefore focus on learning effective behaviour in an offline fashion. Subsequently, in actual play we apply the learned behaviour on the basis of models of the opponent player.

C. Game Environment

To test our approach, we implemented dynamic formations into the Open Real Time Strategy (ORTS) game environment [9]. ORTS, illustrated in Figure 1, is an RTS game environment designed specifically for AI research. ORTS has five properties that make it suitable for AI research, namely (1) it is free software, (2) it uses a flexible game specification, (3) it enables a hack-free game environment, (4) it offers total player control, and (5) custom game AIs can be used [10], [11].

Since 2006, an annual ORTS tournament takes place during the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE). In the tournament, custom built game AIs compete in different game modes. For our experiments, we use the so-called fourth game mode of the annual ORTS tournament. This game mode is focussed on small-scale tactical combat, unit group management, and adversarial/cooperative pathfinding. The game mode's objective is to destroy as many opponent units as possible within a timeframe of five minutes. In practice, the game is finished (i.e., one of the players is annihilated) within these five minutes.

In the fourth game mode, combat takes place as follows. Two players are pitted against each other. Each player controls an army consisting of fifty identical game units. Each unit can fire a shot at an object that is within a predefined range. After firing a shot, a short cooldown period starts during which the unit is not able to fire. The map on which combat takes place is flat. However, the map is populated by moving obstacles, in the form of sheep that roam the map. Perfect information of the game environment is available to all players.

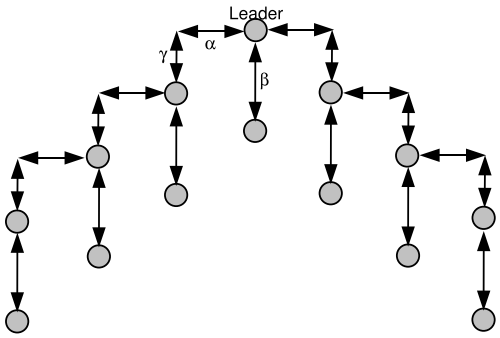


Fig. 2. General design of a dynamic formation.

III. Dynamic Formations

We define a dynamic formation as a group of cooperating units that are capable of adapting to changing circumstances. To obtain effective behaviour, in particular the dynamic formations should be capable of adapting to the formation of the opponent player. To establish such a capability, in our view five aspects are required: (A) a dynamic formation shape (i.e., a non-fixed shape), (B) units in the formation are positioned properly, (C) units in the formation are capable of moving as a group, (D) units in the formation select intelligently which opponent unit to attack, and (E) units in the formation cooperate in their combat behaviour.

In this section we will define an architecture to establish formations that meet these requirements. The architecture allows established formations to be adjusted dynamically by varying formation parameters.

A. Shape of the Formation

In Figure 2, the general design of a dynamic formation is illustrated. The architecture allows for numerous dynamically determined shapes of the formation. For instance, a rectangular shape, a wedge, or a reversed wedge. In our approach, all computer-controlled units are first divided into φ formations. Each formation can have a separate shape, which is determined as follows. A formation is constructed from a grid, arranged in *lines* of positions. The lines in the grid are placed behind each other. Each line consists of a fixed number of units ψ. The units on the first line have a distance α between each other. Any following line will have a distance β behind its predecessor.

The formation is centered around a so-called leader unit, which determines the general direction of movement and the speed of a formation. This leader is a unit that is positioned in the middle of the first line. The units on the left and on the right of the leader are positioned a distance of γ in front, or behind the leader, depending on the value of γ. In Table I, an overview is given of the all parameters to define a formation.

B. Position of the Units

The position of units in the formation is determined according to their initial location on the map. First, units are assigned to the φ formations according to their x-coordinates.

TABLE I

PARAMETERS TO DEFINE THE SHAPE OF A FORMATION.

Parameter		Description
First line distance	α	The distance between units in the first line, where $\alpha \in \{5, \ldots, 50\}$.
Horizontal distance	β	The horizontal distance between different lines in the same group, where $\beta \in \{5, \ldots, 50\}$.
Vertical distance	γ	The vertical distance between the units of neighbouring rows, where $\gamma \in [-50 \ldots 50]$.
Formation speed	δ	The formation speed is the speed at which the formations should manoeuvre, where $\delta \in \{1, 2, 3\}$
Number of formations	φ	The number of different formations the available unit should be organised into, where $\varphi \in \{1, \ldots, 10\}$. Units will be distributed uniformly over the formations wherever possible.
Units per line	ψ	The number of units on each line of the formation, where $\psi \in \{1, \ldots, 50\}$. Note that if there are less units available than the given value, only one line will be created.
Opponent selection		The set of rules that determine the opponent selection behaviour. In our implementation, four opponent selection schemes are available. These are discussed in Subsection III-D.
Combat behaviour		The set of rules that determine the employed combat behaviour. In our implementation, five combat strategies are available. These are discussed in Subsection III-E.

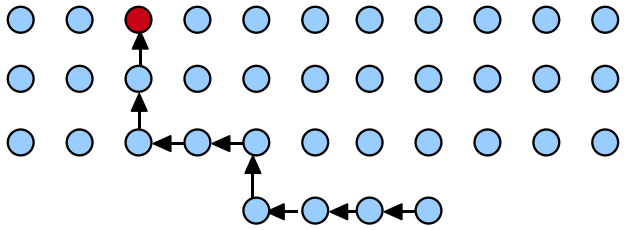


Fig. 3. A destroyed unit from the first line is replaced by a neighbouring unit.

Second, units in each group are distributed over the defined amount of lines, according to their y-coordinates. Third, the position of an individual unit on a line is assigned according to the x-coordinate of this unit. To determine the position of the units via this algorithm is quite efficient, namely $O(n \log n)$, where n is the number of units.

Naturally, we have to consider that during combat, units in the formation may be destroyed. To maintain the shape of the formation, the position of units may need to be adjusted. The focus on adjusting the position of the units lies on replacing units from the first lines (the front), with units from the back. This is done in a computationally inexpensive manner, by only adjusting the position of neighbouring units. This process is illustrated in Figure 3, in which a destroyed unit from the first line is replaced by a neighbouring unit. In practice, this allows the formation to be adjusted rapidly, as units only have be slightly repositioned.

C. Movement of the Formation

A computationally inexpensive approach is used to move the formation as a whole. Our approach to movement of the formation consists of two steps. First, we calculate the direction the leader of the formation should move towards. In our implementation, the leader of a formation will steer towards the nearest formation of the opponent. Second, we set the other units in the formation to follow in the direction parallel to the direction of the leader.

We acknowledge the importance of keeping units in the formation together. It is possible, however, that units may fall out of formation, for instance due to obstacles. To ensure that units remain in formation, in our implementation we temporarily increase the speed of a unit to enable it to catch up with the formation, while we temporarily decrease the speed of the formation as a whole. This is discussed in more detail in previous work [12].

D. Opponent Selection

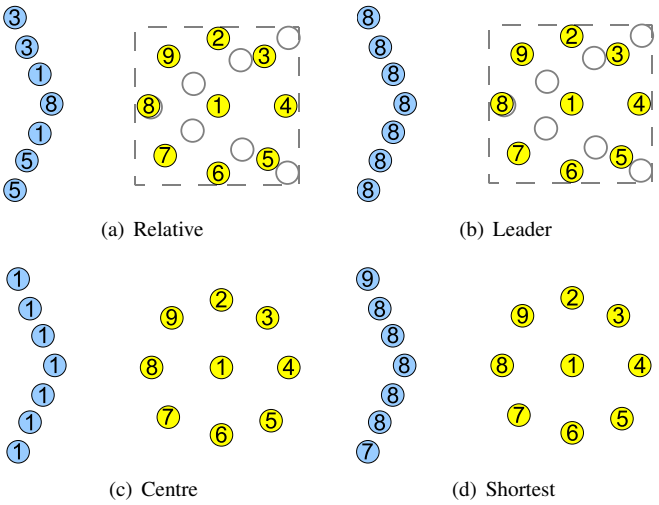
Units in the formation should select intelligently which nearby opponents unit to attack. As a first step to establish such behaviour, a formation is set to attack the most nearby opponent formation. To decide what constitutes an opponent formation is determined by applying a straightforward clustering algorithm. As a second step, individual units in each formation apply rules to select automatically which opponent unit to attack. This is implemented via four different schemes for opponent selection. These four schemes are discussed below.

- **Relative selection:** The scheme for relative opponent selection, illustrated in Figure 4(a), selects for each unit the opponent with the same *relative position* as the concerning unit. This is implemented by mirroring the position of a unit into the formation of the opponent, and subsequently calculating which opponent unit is most near to this mirrored position.
- **Leader selection:** The scheme for leader opponent selection, illustrated in Figure 4(b), is comparable with the scheme for relative selection. However, instead of every unit selecting an opponent unit, only the leader selects an opponent unit. The other units in the formation will subsequently adopt the selection of the leader.
- **Centre selection:** The scheme for centre opponent selection, illustrated in Figure 4(c), selects for each unit the opponent unit that is positioned most nearby the centre of the opponent formation.
- **Nearby selection:** The scheme for nearby opponent selection, illustrated in Figure 4(d), selects for each unit the opponent unit that it is closest to.

E. Combat Behaviour

To determine the behaviour of units in combat with opponent units, we defined five combat strategies. These five strategies are discussed below.

- **Overrun:** The overrun combat behaviour will continuously keep on pushing the units towards the opponent units.



(a) Relative (b) Leader

(c) Centre (d) Shortest

Fig. 4. Opponent selection schemes. The dynamic formation is located on the left, the assigned enemy cluster on the right. The number of the selected opponent is displayed in the unit itself.

- **Hold:** The hold combat behaviour will make the units stop moving as long as the opponent units are within range of their weapons.
- **Retreat:** The retreat combat behaviour is a so-called hit and run strategy, in which the units attack the opponent units, and during the cooldown period retreat continuously.
- **Bounce:** The bounce combat behaviour is similar to the retreat combat strategy. In the bounce combat behaviour, units will attack the opponent, retreat during half of the cooldown period, and in anticipation of the end of the cooldown period, subsequently move towards the opponent.
- **Border:** The border combat behaviour ensures that units are outside the weapons range of the opponent units during the cooldown period. Once positioned outside of the weapons range of the opponent units, the units will stand still until the cooldown period has ended.

IV. Learning which Formation to Employ

To determine automatically which formation to employ in actual play, we incorporate a learning algorithm. The learning algorithm implements stochastic optimisation [13] to learn the best parameters to define a formation. For each parameter, described in Table I, a set of parameter values is available for the learning algorithm to select. The probability of a particular parameter value being selected is determined by the weight assigned to it. Before the learning starts, all weights are assigned a neutral value of one. During play of the game, the weights of parameter values are updated gradually to reflect their actual effectiveness.

The function to update the weights after a game has been played, is denoted by

$$\lambda_v = s\frac{x_n}{\sigma} \tag{1}$$

where λ_v is the update λ given to a particular parameter value v, s is the total number of available values for the concerning parameter, x is a vector storing all game results, x_n is the fitness value of the n–th game, and σ is the learning rate. The calculation of the fitness value x_n, is denoted by

$$x_n = u_f - u_o \tag{2}$$

where u_f is the number of friendly units, and u_o is the number of opponent units.

V. Opponent Modeling

An important factor that influences the effectiveness of a formation, is the formation employed by the opponent. To make predictions about the behaviour of the opponent, an AI player can establish an opponent model. Many researchers point out the importance of modelling the opponent's behaviour [14], [15], [16], [17], [18], [19], and state that opponent models are sorely needed to deal with the complexities of state-of-the-art video games [20].

In general, an opponent model is an abstracted description of a player or of a player's behaviour in a game [20]. Opponent modeling can be seen as a classification problem, where data that is collected during the game is classified as one of the available opponent models. Behaviour of the game AI is established based on this classification of the opponent. This approach is similar to approaches that see known opponent models as stereotypes and an compactification of observations [21].

In this section, we will first discuss how we establish models of the opponent. Second, we discuss how we classify the opponent based on game observations.

A. Modeling the Opponent

We model behaviour of the opponent into a so-called explicit opponent model. An explicit opponent model is a model in which only behavioural features are incorporated, and not the effectiveness of behaviour expressed by these features. This approach to model an opponent has achieved good results in previous research [22].

In our implementation, we model three behavioural features. These features are discussed below.

- **Number of Formations:** This feature reflects into how many formations all units of the opponent have been organised. The feature value is determined by applying a straightforward k-means clustering algorithm.
- **Unit Distribution:** This feature reflects the general distribution of units over the map. The feature value is high if the opponent units are distributed over a narrow but wide area of the map, and the feature value is low if the opponent units are distributed over a thin but high area of the map.
- **Unit Distance:** This feature reflects how densely the opponent units are positioned near each other. The feature value is determined by averaging over the distance between each opponent unit and the most nearby neighbouring unit.

The value of each feature of the opponent model is calculated during the game, just before the units will engage in combat.

B. Classification of the Opponent

The established models are used to classify opponents based on actual game observations. The general procedure is to calculate the likelihood of game observations resulting from each of the established opponent models. The opponent model that best explains game observations is selected.

This classification of the opponent consists of five steps.

1) For each opponent feature, calculate the mean sample value over the total number of observational samples.
2) For each opponent, establish a Gaussian distribution for each feature.
3) Using Bayes' theorem [23], [24], determine for each opponent the likelihood of each feature.
4) Determine the likelihood of each opponent exhibiting the observed feature values. Because the features are independent, the likelihood of each feature for each opponent can be multiplied to determine the combined likelihood of an opponent.
5) For a specific combination of values for each of the features, the opponent is classified according to Bayes' theorem, i.e., by means of normalisation, select the opponent with the highest likelihood.

A detailed description of these five steps is available in previous work [12].

VI. Experiments

This section discusses experiments that test our implementation of dynamic formations in the ORTS game environment. First, we discuss the experiments to test dynamic formations, together with the experimental results. Second, we discuss how well the established opponent models are able to classify the opponent player. Third, we discuss how the established opponent models may be applied to adapt game behaviour.

A. Dynamic Formations

To test our approach, we perform experiments in the ORTS game environment. An experimental run consists of 200 game trials in which two teams play until one team is defeated in combat. After each game trial, the learning mechanism adapts the weights of the formation parameters (see Section IV). We compared several values for the learning rate (a 'normal' learning rate of 200, a 'slow' learning rate of 500, and a variable learning rate based on convergence), and found a learning rate σ of 200 to learn fast and to have a relatively low probability of remaining in a local optimum [12]. In our experiments, we tested a team controlled by the learning mechanism to establish dynamic formations, while in competition with six different, qualitatively well-performing opponents that are developed by several university teams. Five of which where submitted to the ORTS tournament at the AIIDE 2007 conference, and one (UM) was created by us specifically for these experiments. The six opponents are 'Blekinge', 'NUS', 'UBC', 'UM', 'WarsawA' and 'WarsawB'.[1] Each of the two teams starts with fifty identical units. The fitness value that the friendly team can obtain is in the range $\{-50 \ldots 50\}$, a fitness value of 50 being a perfect score (see Eq. (2)). Each experimental run of 200 game trials is performed three times.

To quantify the performance obtained by learning dynamic formations, three properties of an experimental run are used: the absolute performance, the relative performance, and the turning point. We define the absolute performance as the number of games won by the learning team. We define the relative performance as the fitness value obtained by the learning team. We define the turning point as the game trial at which the learning team obtains a win-loss ratio of at least 15 wins against 5 losses in a sliding window of 20. When the ratio is reached, the probability of the learning team outperforming the opponent is larger than 98% [25]. Because of the size of the sliding window, the minimum turning point that can be obtained is 20.

Table II gives an overview of the experimental results. Figure 5 displays the fitness values that are typically obtained by the learning team in competition with the six opponents. We observe that NUS exhibits behaviour that is too strong to be defeated by learning dynamic formations. This comes as no surprise, as NUS won the ORTS 2007 tournament by winning 99% of all played games. Game observations lead us to believe that the low-level AI of NUS is of such an outstanding quality, that learning dynamic formations by itself will not suffice to defeat it.[2] In play against the other five opponents, we observe that in all cases both the absolute performance as well as the relative performance increased or remained excellent while learning. Subsequently, we observe that against all opponents turning points could be achieved, which indicates that behaviour has been learned to significantly outperform the opponent player. In addition, the obtained turning points are relatively low, which indicates that effective dynamic formations are established after a limited amount of learning trials. From these results, we may conclude that our approach allows for successful dynamical formations to be established automatically.

B. Classifying the Opponent

Over the course of all learning trials, feature data is collected that is used to establish the opponent models. Feature data is collected just before combat is about to commence; when the units of the opponent team are still in formation, and the friendly team is still able to change its own formation. Using the procedure denoted in Subsection V-B, we calculate the likelihood of feature values being observed when competing against a particular opponent. This results

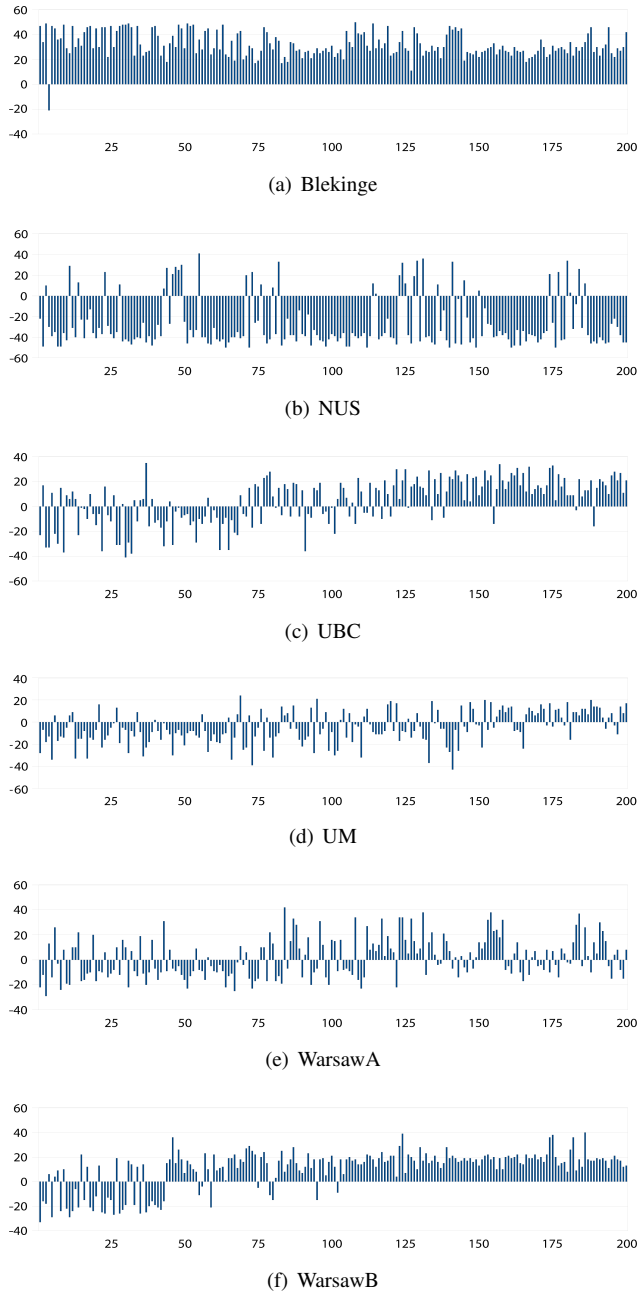
[1] Blekinge was developed by the Blekinge Institute of Technology (Sweden), NUS was developed by the National University of Singapore (Singapore), UBC was developed by the University of British Columbia (Canada), UM was developed by the University of Maastricht (The Netherlands), and WarsawA and WarsawB were developed by the Warsaw University (Poland).

[2] The low-level AI of the NUS opponent optimises player positions by intelligently and continously monitoring player positions, opponent player positions and effective cooldown periods.

TABLE II

RESULTS OF LEARNING DYNAMIC FORMATIONS.

	Abs. perf. (1-50)		Abs. perf. (151-200)		Rel. perf. (1-50)		Rel. perf. (151-200)		Turning point	
	Average	St. dev.	Average	St. dev.	Average	St. dev.	Average	St. dev.	Average	St. dev.
BLEKINGE	48	3	50	0	37.81	7.56	36.57	13.52	22	3
NUS	10	11	6	6	-27.23	26.05	-31.46	21.61	>200[2]	0
UBC	23	9	47	2	-3.83	18.04	15.45	8.76	71	31
UM	9	3	33	7	-11.15	12.20	3.95	11.16	>174	33
WARSAWA	8	5	22	17	-10.44	13.76	-0.95	16.86	>174	46
WARSAWB	21	6	49	1	-5.43	16.81	17.02	5.49	57	12



(a) Blekinge

(b) NUS

(c) UBC

(d) UM

(e) WarsawA

(f) WarsawB

Fig. 5. Fitness values typically obtained while learning dynamic formations.

in three models for opponent classification, one for each feature of the opponent models. Figure 6 illustrates these likelihoods for opponent classification as a function of the observed value for each of the three features of the opponent model. Note that we exclude three of the six opponents in the opponent model, as we will later test the capability of the established opponent models to generalise when confronted with previously unknown opponents.

When classifying the opponents during the course of an ORTS game, already at the first observation the established opponent models are able to classify accurately the opponents (i.e., with a probability that approximates 1.0 in all cases). From these results, we may conclude that known opponents are classified accurately.

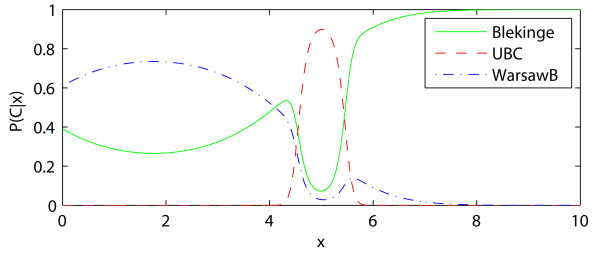
C. Applying Opponent Models

When in competition against a known opponent, i.e., an opponent who's playing features have been captured in the opponent models, we can directly employ successful behaviour that was established by the learning algorithm. Our previous research has shown that when competing with known opponents, applying previously learned behaviour enables a dynamic formation to be effective from the onset of the game [12].

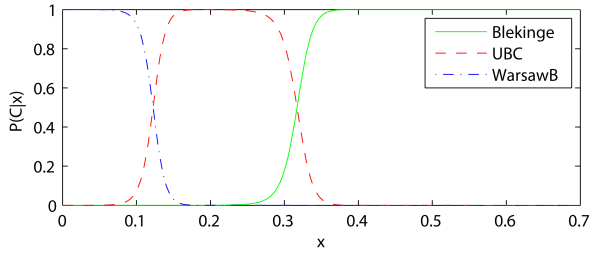
Moreover, when in competition against a previously unknown opponent, i.e., an opponent who's playing features have not been captured in the opponent models, we can still apply the established opponent models for the purpose of improving game behaviour. Namely, unknown opponents may be similar to known opponents with regard to their playing features. It may therefore be expected that behaviour that is successful against one particular opponent, will also be successful against similar opponents.

In our experiments we observe that the three opponents which were excluded from the opponent models, all are classified as most similar to WarsawB. Table III gives an overview of the experimental results of applying the behaviour learned in play against WarsawB to the three unknown opponents. The results reveal that when competing with UM and WarsawA, in all cases highly effective formations are established instantly. It may seem that no

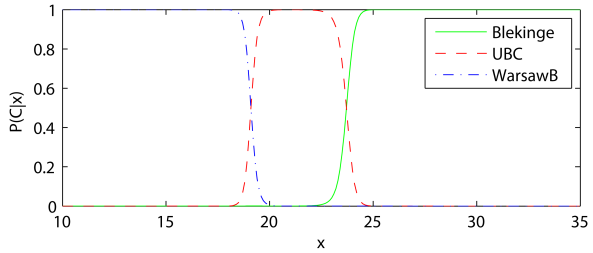
[2]With the '>' symbol we indicate that in at least one run against the concerning opponent, a turning point was not reached before the learning ended. In the results this run was processed as having at least the maximum turning point (200). Therefore, the average turning point over all runs will be at least the given value.



(a) Number of Formations



(b) Unit Distribution



(c) Unit Distance

Fig. 6. Likelihoods of the three features of the opponent model.

learning of formations is taking place, however the turning indicates that some learning takes place, though already very effective behaviour is established at the start. In addition, the effectiveness of the established dynamic formations does not degrade during play of the game. These results illustrate that by incorporating opponent models in a game's AI, behaviour can be established that is and remains effective from the onset of a game.

Naturally, situations may arise where learned behaviour is ineffective when applied in competition with previously unknown opponents. As Table III also reveals, this was the case in the trails against NUS, where directly applying the learned behaviour was not sufficient to be effective against NUS. We believe this is a particular case which illustrates that it is not possible to defeat NUS with only dynamic formations at hand. Rather, ways to defeat NUS are likely to be found in the domain of low-level AI. In typical cases, on the other hand, ineffective behaviour is bound to result from previously unknown opponents exhibiting playing features that have not yet been captured in the established opponent models. As a result, learned behaviour should not be applied directly, but should be adapted according to the new circumstances. This will be the focus of future research.

VII. Conclusions and Future Work

In this paper we discussed an approach to establish dynamic formations. In the approach, dynamic formations are established via a learning algorithm. Subsequently, models of the opponent player are created and utilised for the purpose of applying the learned behaviour. Results of experiments that test the approach in the ORTS game environment show that behaviour is learned that is effective against five of the six employed opponents. Only the strong NUS opponent exhibited such strong low-level AI that it could not be defeated by dynamic formations alone. The results reveal that the established opponent models can accurately classify an opponent player. This classification, in turn, has been used to directly apply successful behaviour, in order to create AI that is effective from the onset of a game. Because the resulting game adaptation was effective, as well as efficient (when using opponent models), we may conclude that the established approach to dynamic formations can be successfully incorporated in an actual video-game environment.

For future work, we will investigate how our approach to apply established opponent models can be extended to allow for automatically adapting learned behaviour to play of previously unknown opponents.

Appendix

In this appendix, we describe the learned formations that are employed when competing with the known opponents Blekinge, UBC and WarsawB.

- **Blekinge:** First line distance: 35. Horizontal distance: 15. Vertical distance: 6. Formation speed: 1. Number of formations: 5. Units per line: 25. Opponent selection: Centre. Combat behaviour: Bounce.
- **UBC:** First line distance: 25. Horizontal distance: 20. Vertical distance: 8. Formation speed: 1. Number of formations: 8. Units per line: 25. Opponent selection: Centre. Combat behaviour: Bounce.
- **WarsawB:** First line distance: 30. Horizontal distance: 10. Vertical distance: 2. Formation speed: 1. Number of formations: 5. Units per line: 50. Opponent selection: Nearby. Combat behaviour: Retreat.

Acknowledgements

This research is funded by a grant from the Netherlands Organization for Scientific Research (NWO grant No 612.066.406) and is performed in the framework of the ROLEC project.

References

[1] C. Dawson, *AI Game Programming Wisdom*. Charles River Media, Inc., 2002, ch. Formations, pp. 272–281, ISBN 1-584-500778.

[2] W. Trumble and L. Brown, "Shorter Oxford English Dictionary," 2002, fifth Edition, Oxford University Press, USA, 2002.

[3] C. Lieut.-Col. Gudgeon, "Maori wars," *Journal of the Polynesian Society*, vol. 16(1), pp. 13–42, 1907.

[4] I. Millington, *Artificial Intelligence for Games*. San Francisco, California: Morgan Kaufmann Publishers Inc., 2006, ch. Coordinated Movement, pp. 151–177, ISBN 0-124-977820.

[5] P. Demasi and A. J. de O. Cruz, "Online coevolution for action games," *International Journal of Intelligent Games and Simulation*, vol. 2(3), pp. 80–88, 2002.

TABLE III

RESULTS OF APPLYING OPPONENT MODELS TO UNKNOWN OPPONENTS.

	Abs. perf. (1-50)		Abs. perf. (151-200)		Rel. perf. (1-50)		Rel. perf. (151-200)		Turning point	
	Average	St. dev.	Average	St. dev.	Average	St. dev.	Average	St. dev.	Average	St. dev.
NUS	0	1	2	2	-39.49	8.8	-37.45	14.32	>200	0
UM	40	1	42	4	7.94	8.61	8.91	8.28	23	5
WARSAWA	34	2	33	5	5.16	9.03	4.79	9.02	22	2

[6] S. Johnson, *AI Game Programming Wisdom 2*. Charles River Media, Inc., Hingham, MA, 2004, ch. Adaptive AI: A Practical Example, pp. 639–647.

[7] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Machine Learning*, vol. 63(3), pp. 217–248, 2006.

[8] S. Rabin, *AI Game Programming Wisdom 4*. Charles River Media, Inc., 2008, ch. Preface - What happened to learning?, pp. ix – xi, ISBN 1-584-505230.

[9] M. Buro, T. Furtak, A. Kovarsky, M. Lanctot, and S. Orsten, "The open real-time strategy (ORTS) programming environment for RTS games," 2008, [Online]. Available: http://www.cs.ualberta.ca/~mburo/orts/

[10] M. Buro and T. Furtak, "RTS games as test-bed for real-time research," in *Invited Paper at the Workshop on Game AI, JCIS*, Alberta, Canada, 2003, pp. 481–484.

[11] M. Buro, "ORTS: A hack-free RTS game environment," in *Proceedings of the International Computers and Games Conference*, J. Schaeffer, M. Müller, and Y. Björnsson, Eds. Alberta, Canada: Springer, 2002, pp. 280–291.

[12] M. van der Heijden, "Dynamic formations in real-time strategy games," 2008, masters thesis, Department of Computer Science, Universiteit Maastricht, The Netherlands. [Online]. Available: http://ticc.uvt.nl/~pspronck/theses/Thesis_Heijden.pdf

[13] J. C. Spall, "Stochastic optimization," in *Handbook of Computational Statistics*. Springer, Berlin, 2004, pp. 169–199, ISBN 3-540-404643.

[14] B. Abramson, "Expected outcome: A general model of static evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 182–193, March 1990.

[15] H. Berliner, "Search and knowledge," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 77)*, 1977, pp. 975–979.

[16] D. Knuth and R. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6(4), pp. 293–326, 1975.

[17] R. Korf, "Generalized game trees," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 89)*. Detroit, MI, August 1989, pp. 328–333.

[18] A. Samuel, "Some studies in machine learning using the game of chechers, ii - recent progres," *IBM Journal*, vol. 11, pp. 601–617, 1967.

[19] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, "A world championship caliber checkers program," *Artificial Intelligence*, vol. 53, pp. 273–289, 1992.

[20] J. van den Herik, J. Donkers, and P. Spronck, "Opponent modelling and commercial games," in *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (eds. Graham Kendall and Simon Lucas)*, 2005, pp. 15–25.

[21] J. Denzinger and J. Hamdan, "Improving modeling of other agents using tentative stereotypes and compactification of observations," in *Proceedings of the International Conference on Intelligent Agent Technology (IAT)*, 2004, pp. 106–112.

[22] D. Carmel and S. Markovitch, "Exploration and adaptation in multiagent systems: A model-based approach," in *Proceedings of The Fifteenth International Joint Conference for Artificial Intelligence*, Nagoya, Japan, 1997, pp. 606–611.

[23] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, Massachusetts: MIT Press, October 2004, ISBN 0-262-012111.

[24] C. Scientific, "Elements of Visual Statistics," 2008, retrieved May 27, 2008. [Online]. Available: http://www.visualstatistics.net/

[25] P. R. Cohen, *Emperical Methods for Artificial Intelligence*. The MIT Press, 1995, ISBN 0-262-032252.