

Validation of Agent Based Distillation Movement Algorithms

Andrew W Gill¹ and Dion Grieger²

**1. Military Systems Experimentation Branch
2. Land Operations Division
Electronics and Surveillance Research Laboratory**

DSTO-TN-XXXX

ABSTRACT

Agent based distillations (ABD's) are low-resolution abstract models, which can be used to explore questions associated with land combat operations in a short period of time. Movement of agents within the EINSTEIN and MANA ABD's is based on a simple attraction-repulsion weighting system and an associated numerical penalty function. The relative simplicity of these ABD's seems to have led to the general acceptance of their associated movement algorithms, without much validation or analysis of behaviour. This paper analyses these movement algorithms and finds unwanted behaviour and proposes suggestions for improvement to the penalty function based on relative distances. A more novel technique based on the concepts underlying spatial estimation is also proposed as an alternative.

RELEASE LIMITATION

Approved for public release

Published by

*DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108 Australia*

*Telephone: (08) 8259 5555
Fax: (08) 8259 6567*

*© Commonwealth of Australia 2001
AR- XXX-XXX
December 2001*

Conditions of Release and Disposal

This document is the property of the Australian Government; the information it contains is released for defence purposes only and must not be disseminated beyond the stated distribution without prior approval.

The document and the information it contains must be handled in accordance with security regulations applying in the country of lodgement, downgrading instructions must be observed and delimitation is only with the specific approval of the Releasing Authority as given in the Secondary Distribution statement.

This information may be subject to privately owned rights.

The officer in possession of this document is responsible for its safe custody. When no longer required DSTO Reports should be returned to the DSTO Library, (Reports Section), Salisbury SA.

Validation of Agent Based Distillation Movement Algorithms

Executive Summary

Based on counter-intuitive results from a recent study conducted by the authors using the EINSTEIN ABD, the movement algorithms of that ABD and the MANA ABD were investigated.

A simplified one-dimensional scenario was used to deduce the causes for the unwanted behaviour and some simple analysis led to a basic suggestion for improvement to the penalty function based on relative distances. A more novel technique based on the concepts underlying spatial estimation was also proposed as an alternative.

All of these techniques were then compared on both the simplified one-dimensional scenario as well as a static two-dimensional scenario. The results show that these penalty functions do not suffer from the counter-intuitive behaviours that appear to limit the validity of the EINSTEIN and MANA penalty functions.

However, of these new penalty functions there has been no attempt to suggest which one is 'correct' or 'best', as this would be a futile exercise. The intent of this paper has been to illustrate the variability of movement paths that the alternative penalty functions can generate.

Having achieved this, we then suggest that the correct approach is to make the penalty function a user-defined 'parameter', in exactly the same way that the entity capabilities and personalities are. At a minimum, it would make more explicit the assumptions made about the movement algorithm when one provides the results or conclusions of a study. Ideally, the robustness of these conclusions should be tested to variations in the movement algorithm, in the same way that sensitivity analysis is applied to other more traditional parameters.

Some improvements to the current implementation of the meta-personalities were also made to improve their flexibility, and suggestions for extending the penalty function to incorporate non-linear utility and stochastic movement were also provided.

The authors hope that any future versions of EINSTEIN, MANA or other new ABD take into consideration the points raised in this paper.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA				1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)	
2. TITLE Validation of Agent Based Distillation Movement Algorithms			3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U)(L) Title (U) Abstract (U)		
4. AUTHOR(S) Andrew W Gill and Dion Grieger			5. CORPORATE AUTHOR Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108 Australia		
6a. DSTO NUMBER DSTO-TN-XXXX		6b. AR NUMBER AR- XXX-XXX	6c. TYPE OF REPORT Technical Note		7. DOCUMENT DATE December 2001
8. FILE NUMBER eg: 510/207/0128	9. TASK NUMBER Task Number	10. TASK SPONSOR Task Sponsor	11. NO. OF PAGES 31		12. NO. OF REFERENCES 8
13. DOWNGRADING/DELIMITING INSTRUCTIONS To be reviewed three years after date of publication			14. RELEASE AUTHORITY Chief, , Land Operations Division		
15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT <i>Distribution additional to the initial list is limited to Australian Department of Defence and Defence Force personnel who are suitably security cleared and have a need-to-know. Others inquiring must be referred to Head of Military Systems Experimentation Branch ESRL.</i> OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, SALISBURY, SA 5108					
16. DELIBERATE ANNOUNCEMENT Australian Department of Defence and Defence Force Personnel.					
17. CASUAL ANNOUNCEMENT Yes					
18. DEFTEST DESCRIPTORS See your Client Liaison Librarian for DEFTEST terms					
19. ABSTRACT Agent based distillations (ABD's) are low-resolution abstract models, which can be used to explore questions associated with land combat operations in a short period of time. Movement of agents within the EINSTEIN and MANA ABD's is based on a simple attraction-repulsion weighting system and an associated numerical penalty function. The relative simplicity of these ABD's seems to have led to the general acceptance of their associated movement algorithms, without much validation or analysis of behaviour. This paper analyses these movement algorithms and finds unwanted behaviour and proposes suggestions for improvement to the penalty function based on relative distances. A more novel technique based on the concepts underlying spatial estimation is also proposed as an alternative.					

Contents

1. INTRODUCTION	1
1.1 Background	1
1.2 Movement within ABD's	1
1.3 Motivation and Scope	2
2. EINSTEIN AND MANA MOVEMENT ALGORITHMS	3
2.1 Examples of Unwanted Behaviour	3
2.2 Simplified One-Dimensional Scenario Analysis	5
2.2.1 Penalty Calculations Using EINSTEIN Movement Algorithm	6
2.2.2 Penalty Calculations Using Mana Movement Algorithm	7
2.3 Discussion of Behaviour	8
3. ALTERNATIVE MOVEMENT ALGORITHMS	9
3.1 Dimensional Analysis and Relative Distance	9
3.2 Cumulative Penalties	10
3.3 The Flag	11
3.4 Spatial Interpolation Technique	13
3.5 Simplified One-Dimensional Scenario Results	15
3.6 Two-Dimensional Scenario Results	16
4. OTHER CONSIDERATIONS	20
4.1 Meta-Personalities	20
4.1.1 Cluster and Advance Meta-Personalities.....	20
4.1.2 Combat Meta-Personality.....	21
4.1.3 MANA Trigger States	22
4.2 Utility Curves	22
4.3 Stochastic Movement	23
5. SUMMARY 23	
6. REFERENCES	24
APPENDIX A: ISAAC MOVEMENT ALGORITHM	26
APPENDIX B: MANA MOVEMENT ALGORITHM	29

1. Introduction

1.1 Background

Agent based distillations (ABD's) are low-resolution abstract models, used to explore questions associated with land combat operations in a short period of time. Being agent based means that only simple behavioural rules need to be assigned. This is generally achieved by assigning 'personalities' to the agents by way of relative weightings to various elements on the battlefield (friendly and enemy agents, notional 'flags', terrain features, etc) and a linear penalty function to determine the entity's next move. Various 'meta-personalities' can also be assigned which moderate the agent's default personality if certain threshold constraints are exceeded from time to time.

Project Albert is a United States Marine Corps (USMC) research effort that aims to identify emergent behaviour through the application of ABD's and seeks to address the areas of non-linear behaviour (where small changes create disproportionate responses); co-evolving landscapes (which characterise the changing battlefield) and intangibles (such as morale, discipline and training) for which conventional land combat analysis models are particularly poor at investigating.

There are a growing number of ABD's under Project Albert, including the Irreducible Semi-Autonomous Adaptive Combat (ISAAC) model [1] and the Enhanced ISAAC Neural Simulation Toolkit (EINSTEIN) [1]. The NZ DTA has also recently developed the Map Aware Non-uniform Automata (MANA), to support their studies [2].

1.2 Movement within ABD's

The User Manual of MANA [3] states that:

"The most important action of an agent is to move."

This appears justified since being deliberately low-resolution means that the detailed physics of combat are largely ignored (or abstracted to simple constructs) and thus any emergent or interesting behaviour should appear as a result of the manoeuvring of the agents (in space and time) about the battlefield.

Movement of agents within EINSTEIN and MANA ABD's is based on a simple attraction-repulsion weighting system and an associated numerical penalty function. From its current location, the agent moves to the location within its movement range that incurs the least penalty. That is, the agent attempts to satisfy its personality-driven desire to move closer to or further away from other agents and either of the two flags. This algorithm is applied to each agent on both sides and each is moved to its new location. This process is repeated for each time step in the simulation.

The form of the penalty function implemented by both the EINSTEIN and MANA ABD's is hard-coded. The user only has control over the value of the weightings towards the agents and flags. The user defines these weightings when a scenario is constructed and is chosen to represent surrogates for the tactics employed by the entities. For example, the EINSTEIN User manual [4] provides examples of aggressive (defensive) postures by assigning relatively large positive (negative) weights to enemy agents.

Given the simplistic nature of the attraction-repulsion weighting system, this will be at best an approximation to the true behaviour of the entity being modelled. What is important then, is that the movement algorithm implements faithfully the relationships that the user believes he is modelling by assigning values to those weights.

For example, consider the situation with a weighting of +40 towards allied entities, a weighting of -10 towards enemy entities, and a weighting of +20 towards the enemy flag. The most natural interpretation of this situation is that the entity is four times more likely to move towards other allied entities than it is to move away from enemy entities, but that it is only two times more likely to move towards other allied entities than it is to move towards the enemy flag.

However, there are two key factors which are not stated in the above interpretation that are important in terms of what the user believes is being modelled. These factors are the number of entities the agent is aware of (generally those within its sensor range) and the distances those entities are from the agent in question.

For example, does the above weighting system interpret the total weight for five enemy entities as -50, or is it independent of the number of enemy entities observed (or is it some non-linear function)?

Similarly, does the above weighting system degrade the weight towards enemy entities as their distance from the agent in question increases (and is this degradation a linear function), or is it independent of these distances?

Both of these questions are important for any weighting system in general, but are particularly relevant for the systems that reside in EINSTEIN and MANA.

1.3 Motivation and Scope

The relative simplicity of ABD's, in particular EINSTEIN and MANA, seems to have led to the general acceptance of their associated movement algorithms, without apparently much validation or analysis of behaviour.

However, counter-intuitive results from a recent study by the authors using the EINSTEIN ABD [5] suggested that the behaviours produced by its movement algorithm may not always be desirable or indeed what the user intended.

This paper examines in detail the movement algorithms of two of the most popular ABD's, EINSTEIN and MANA. Scenarios are presented in Section 2 that highlights quite clearly forms of unwanted behaviour. A simplified one-dimensional scenario is examined in detail to illustrate the two movement algorithms and to discover the causes of the unwanted behaviour.

Two alternative movement algorithms (with variants) are then presented in Section 3 as potential remedies. The first is a natural modification to both the EINSTEIN and MANA movement algorithms while the second is somewhat novel and is based on techniques for spatial interpolation. These alternatives are tested on the one-dimensional scenario and are compared on a simplified (static) two-dimensional scenario.

Section 4 then presents some comments on the meta-personality movement modifiers currently implemented in EINSTEIN and MANA as well as some thoughts on extensions to these penalty functions. Finally, some final thoughts are provided in Section 5.

2. EINSTEIN and MANA Movement Algorithms

2.1 Examples of Unwanted Behaviour

Figures 1 and 2 below illustrate the counter-intuitive behaviour mentioned above. Red's goal is to avoid Blue and get to the flag. Red are repelled from Blue (-20) and attracted to the flag (+100). Blue are attracted to Red (+50). In the first scenario Red has a sensor range of 10 and many agents achieve the goal and make it to the flag (mainly by skirting around the Blue group). The second scenario is identical to the first except that Red's sensor range is increased to 20. However, despite their increased knowledge of the positions of Blue they now move directly into the position of the Blue group and are completely destroyed.

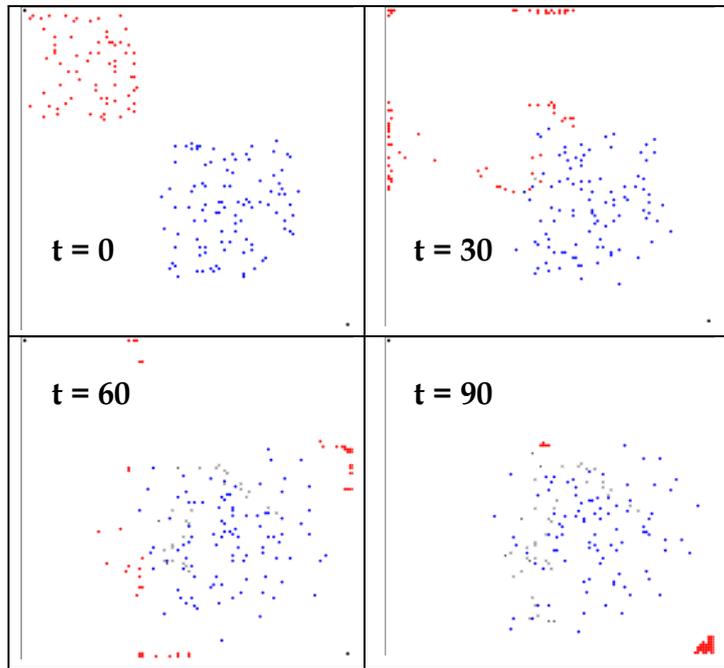


Figure 1 Example of Counter-Intuitive EINSTEIn Behaviour -- Sensor Range = 10

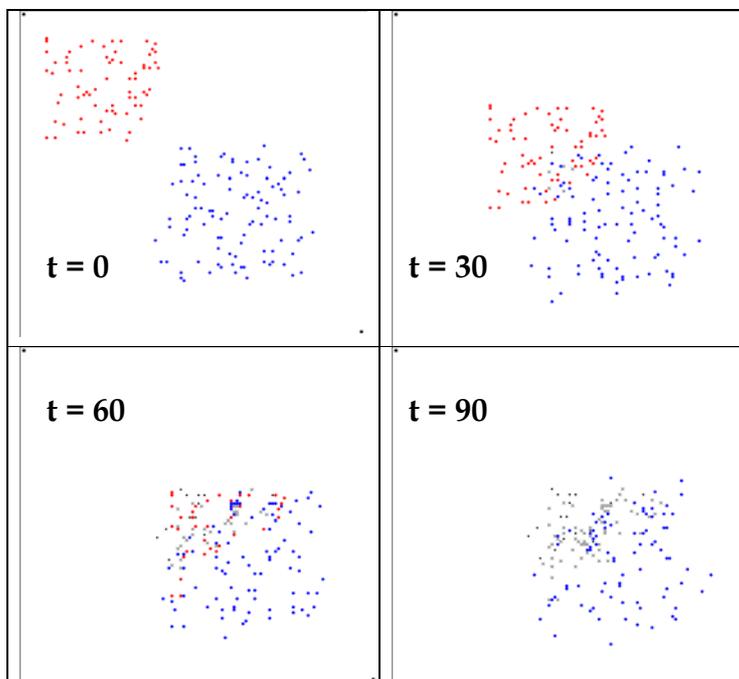


Figure 2 Example of Counter-Intuitive EINSTEIn Behaviour -- Sensor Range = 20

Figure 3 illustrates a MANA scenario where the goal is for the Blue agent to reach the Red flag (weighting = +20) while avoiding Red (weighting = -10). Note that the Blue agent decides to go straight through the group of Red regardless of how many Red agents it sees (no attrition occurs as firepower is set to zero).

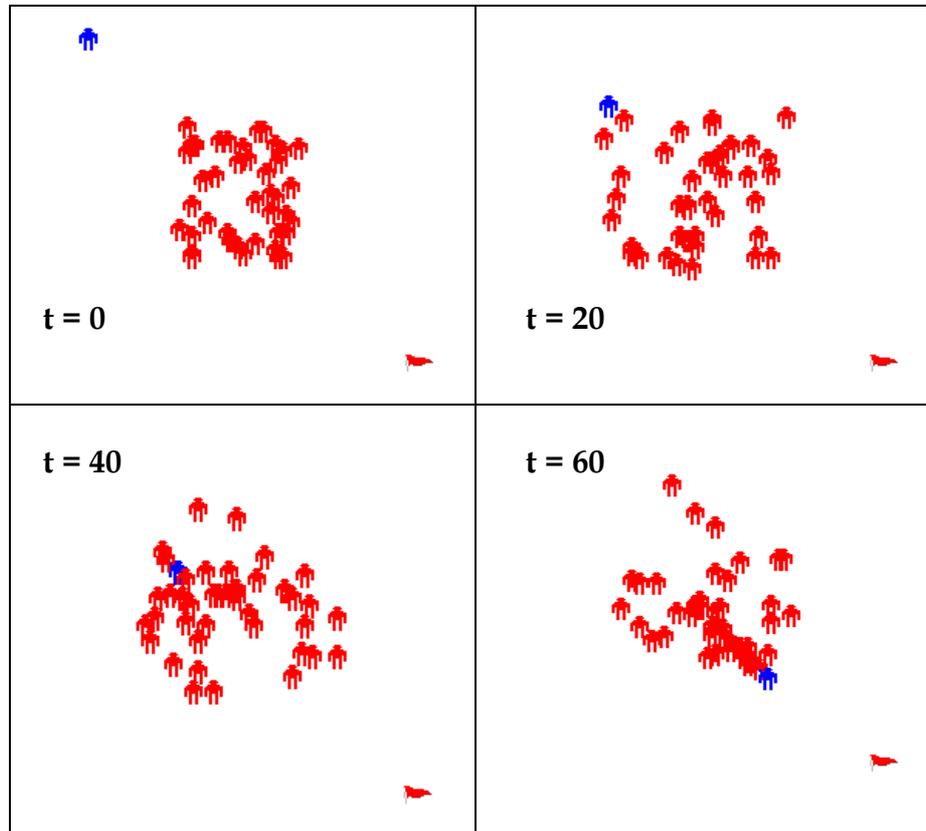


Figure 3 Example of Counter-Intuitive MANA Behaviour

2.2 Simplified One-Dimensional Scenario Analysis

To illustrate the counter-intuitive behaviour than EINSTEIN and MANA can produce, a simple scenario is presented in Figure 1 below to allow the penalty calculations to be explicitly performed.

To explore the reasons for this strange behaviour a simple scenario is presented in Figure 4 below to allow the penalty calculations to be explicitly performed. The single Blue agent is repelled by the Red agents (with weighting -10) and attracted to the flag (with weighting +20). Thus, there is a tension between wanting to move forwards towards the flag and remaining in place to avoid the Red agents. The decision that the Blue agent must make is whether to stay in place (remain at location 1) or move forwards (move to location 2).

										
1	2	3	4	5	6	7	8	9	10	11

Figure 4 Simplified One-Dimensional Scenario

We will examine this scenario with two variants, one where Blue has a sensor range of three, (that is, it can only see the first Red entity), and another where Blue has a sensor range of six (and can thus see both Red entities). Each penalty calculation below shows the enemy and flag components separately, followed by the total penalty. The equations for the two algorithms are also given although restricted to the parts relevant to this scenario (the enemy and flag components).

2.2.1 Penalty Calculations Using EINSTEIN Movement Algorithm

The equation that EINSTEIN uses to compute the penalty at each potential new location is given by

$$Z_{new} = \left(\frac{W_E}{E * R_S \sqrt{2}} \sum_{i=1}^E D_{i, new} \right) + W_F \left(\frac{D_{F, new}}{D_{F, old}} \right). \quad (1)$$

Appendix A contains a direct extract from the EINSTEIN User Manual that details more information about its movement algorithm. Table 1 lists each of the variables in this equation together with their definitions.

Variable	Definition
R_S	Sensor range of agent about to move
E	Number of enemy entities within sensor range
W_E	Weighting towards enemy agents
$D_{i, new}$	Distance to the i th enemy from the new location
W_F	Weighting towards the flag
$D_{F, new}$	Distance to the flag from the new location
$D_{F, old}$	Distance to the flag from the current (old) location

Table 1 Variables Associated with the EINSTEIN Penalty Function

Table 2 tabulates the two components and the total penalty for the choices of staying in place (location 1) or moving forwards (location 2), for the cases where the sensor range is either 3 or 6.

Sensor Range	Location	Enemy Component	Flag Component	Penalty
$R_S = 3$	Current (1)	-7.07	20.00	12.93
	New (2)	-4.71	18.00	13.29
$R_S = 6$	Current (1)	-5.30	20.00	14.70
	New (2)	-4.12	18.00	13.88

Table 2 EINSTEIN Penalty Function Component Calculations

For the case where the sensor range is 3 (thus can only see the first Red agent), the Blue agent would decide to stay in place (since the penalty function is being minimized). However, for the case where the sensor range is 6 (and can thus see both Red agents), the Blue agent apparently decides to move forward. This behaviour is counter-intuitive and will be discussed in the next section.

2.2.2 Penalty Calculations Using Mana Movement Algorithm

The equation that MANA uses to compute the penalty at each potential new location is

$$Z_{new} = \left(\frac{W_E}{100 * E} \right) \left(\sum_{i=1}^E \frac{D_{i,new} + (100 - D_{i,old})}{100} \right) + \left(\frac{W_F}{100} \right) \left(\frac{D_{F,new} + (100 - D_{F,old})}{100} \right) \quad (2)$$

Appendix B contains a direct extract from the MANA User Manual that details more information about its movement algorithm. Table 3 lists each of the variables in this equation together with their definitions.

Variable	Definition
E	Number of enemy entities within sensor range
W_E	Weighting towards enemy agents
$D_{i,new}$	Distance to the i th enemy from the new location
$D_{i,old}$	Distance to the i th enemy from the current (old) location
W_F	Weighting towards the flag
$D_{F,new}$	Distance to the flag from the new location
$D_{F,old}$	Distance to the flag from the current (old) location

Table 3 Variables Associated with the MANA Penalty Function

Table 4 tabulates the two components and the total penalty for the choices of staying in place (location 1) or moving forwards (location 2), for the cases where the sensor range is either 3 or 6.

Sensor Range	Location	Enemy Component	Flag Component	Penalty
$R_S = 3$	Current (1)	-0.10	0.20	0.10
	New (2)	-0.099	0.198	0.099
$R_S = 6$	Current (1)	-0.10	0.20	0.10
	New (2)	-0.099	0.198	0.099

Table 4 MANA Penalty Function Component Calculations

Here, for both cases (sensor range of 3 or 6) the Blue agent decides to move forward. This is different behaviour to that produced by EINSTEIN above. Furthermore, we note that the penalties do not change at all when the Blue agent can see both Red agents. This behaviour is also counter-intuitive and will be discussed next.

2.3 Discussion of Behaviour

Both algorithms above use scale factors in their penalty calculations. EINSTEIN uses the relative distance to the flag $D_{F,new} / D_{F,old}$ but then scales the average distance to other entities (allies or enemy) by $R_S\sqrt{2}$. It is not clear to the authors the rationale for this choice of scaling. It does, to an extent, scale the new distances relative to the old distances and to the relative distances used for the flag. However, as we have seen above it doesn't solve the problem entirely, and as will be seen below there is a simpler solution.

EINSTEIN also scales the summation term by the number of entities (that is, it calculates the average distance). In effect, this implies that the Blue agent only effectively observes one Red entity when deciding on which move to make (and this one entity is positioned at the centroid of the Red entities within sensor range).

These observations explain why the Blue agent remains in place for a sensor range of 3 (because it "sees" one Red entity 3 units away) but moves forward for a sensor range of 6 (because it "sees" only one Red entity 4.5 units away, when in fact there are two Red entities at 3 and 6 units away respectively). This again seems to be undesirable behaviour and not what the user would have intended when the original weights were entered.

MANA does not use relative distance scaling for the flag or other entities in the fashion used by EINSTEIN. Instead, it scales all distances by the constant 100. That is, it treats all entities as if they were 100 units away. This means that the penalty for moving towards a Red that is 5 units away will be the same as the penalty for moving towards a Red that is 50 units away. Again, it appears to the authors that this choice is both too arbitrary and unnecessary. MANA also uses the average distance concept and thus also falls victim to the 'centroid' problem.

It is clear from this simple example that there are two factors in the penalty functions used by EINSTEIN and MANA which are fundamental in terms of implementing what the user believes is being modelled when attraction-repulsion weights are prescribed. These factors are the number of entities the agent is aware of and the distances those entities are from the agent in question.

3. Alternative Movement Algorithms

3.1 Dimensional Analysis and Relative Distance

In the simple one-dimensional scenario above with a sensor range of three, Blue will choose to move forward if the absolute difference in the flag components between Z_1 and Z_2 is greater than the absolute difference in the enemy components between Z_1 and Z_2 . These two quantities are

$$\Delta Z_{Flag} = \frac{W_F * D_F}{D_F} - \frac{W_F(D_F - 1)}{D_F} = \frac{W_F}{D_F}$$

and

$$\Delta Z_{Enemy} = \frac{W_E * (D_E - 1)}{R_S \sqrt{2}} - \frac{W_E * D_E}{R_S \sqrt{2}} = -\frac{W_E}{R_S \sqrt{2}}$$

where D_F and D_E are the distances to the flag and to the Red entity from the original location, respectively. Thus, Blue will choose to move forward when

$$\frac{W_F}{D_F} > -\frac{W_E}{R_S \sqrt{2}} \Rightarrow -\frac{W_F}{W_E} > \frac{D_F}{R_S \sqrt{2}}.$$

The left hand side of this inequality implies that the weights obey a proportional or relative law. That is, the ratio of the weights must exceed a certain threshold in order to affect a move forward. This threshold is the left hand side of the inequality and the first point to note is that is independent of D_E the distance to the enemy, but does depend on the unusual scaling factor discussed above. A similar analysis for the MANA penalty function generates similar findings.

The most obvious and natural modification to the inequality above is

$$-\frac{W_F}{W_E} > \frac{D_F}{D_E}$$

which implies that Blue will move forward provided the attraction towards the flag exceeds the repulsion from the enemy by more than the distance to the flag exceeds the distance to the enemy.

Generalising this, an alternative movement algorithm is proposed whereby the denominator of the enemy (and ally) components of the penalty are divided (or scaled) by $D_{i,old}$ (the distance from the current location to the location of the enemy). This would replace the artificial and arbitrary scaling factors of EINSTEIN and MANA (sensor range and the constant 100).

The proposed alternative penalty function is then given by

$$Z_{new} = \left(\frac{W_E}{E} \sum_{i=1}^E \frac{D_{i,new} - D_{i,old}}{D_{i,old}} \right) + W_F \left(\frac{D_{F,new} - D_{F,old}}{D_{F,old}} \right). \quad (3)$$

This equation is dimensionally correct and using the absolute difference of distances in the numerators and the old distances in the denominator means that the change in distances (from the current location to the new location) are compared, and compared in a relative sense. An added bonus is that the penalty for staying in place is by definition equal to zero and thus needn't be calculated (thus saving some computation time).

The use of relative distances in this fashion appears to solve the scaling problem inherent in both EINSTEIN and MANA. Under this new penalty function, entities would then assign a stronger weight to those entities that are nearby than that to those far away. For example, in the simplified one-dimensional scenario above, when the Blue entity moves forward it is 33% closer to the first Red entity, only 17% closer to the second Red entity, and only 10% closer to the flag, and these relative percentages will be reflected under the new penalty function.

3.2 Cumulative Penalties

However, as presented above, this new penalty function still computes an average (relative) distance as it divides by the number of enemy within sensor range, and thus subject to the 'centroid' issue discussed above. It appears to the authors that a more usual form of the penalty calculation would use a cumulative function instead of an average. This would involve simply removing the denominator at the front of the previous equation, thus the alternative penalty function is then given by

$$Z_{new} = \left(W_E \sum_{i=1}^E \frac{D_{i,new} - D_{i,old}}{D_{i,old}} \right) + W_F \left(\frac{D_{F,new} - D_{F,old}}{D_{F,old}} \right). \quad (4)$$

However, at times it may not be desirable to use either the cumulative or average functional. A generalisation of the above penalty function to incorporate this is given simply by

$$Z_{new} = \left(\frac{W_E}{E^\alpha} \sum_{i=1}^E \frac{D_{i,new} - D_{i,old}}{D_{i,old}} \right) + W_F \left(\frac{D_{F,new} - D_{F,old}}{D_{F,old}} \right) \quad (5)$$

where α is a real value between zero and one. The cases $\alpha=0$ and $\alpha=1$ reduce to equations (4) and (3), respectively. Thus as α increases from zero the penalty function moves from using cumulative distances towards using the average distance. However, moving away from using an average functional introduces issues regarding the flag that must be investigated.

3.3 The Flag

The problem is not so obvious with this simple scenario but it is apparent through examining the equations that for lower values of α and larger values of E the flag component of the penalty will become increasingly insignificant. Intuitively this may not be a problem because it seems logical that one should be more concerned with a large number of enemies than with a single flag some distance away. For this reason the current form of the penalty function was still investigated along with some additional forms.

These additional forms attempt to allow the entities to scale the weight to the flag in the same way that the weight to the enemies (and allies) is scaled. Four alternatives for the flag component of the penalty are listed below along with a description of what effect they are attempting to achieve. Note that A is defined as the number of allies, or Blue agents.

The first alternative is the default case, as given in equation (5)

$$Z_{F1} = W_F \left(\frac{D_{F,new} - D_{F,old}}{D_{F,old}} \right) \quad (6)$$

and is given here explicitly only to allow comparison with the other alternatives. This form assumes that there is only one flag and the scaling factor is the distance to the current location to the flag.

The second alternative is given by

$$Z_{F2} = W_F \left(\frac{D_{F, new} - D_{F, old}}{\frac{1}{E} \sum_{i=1}^E D_{i, old} + \frac{1}{A} \sum_{i=1}^A D_{i, old}} \right) \quad (7)$$

which assumes that there is only one flag but uses a scaling factor which is the average distance of all detected enemy and ally entities. The effect of this in the simple scenario is that the flag is in essence now located between locations five and six, as illustrated in Figure 5 below.

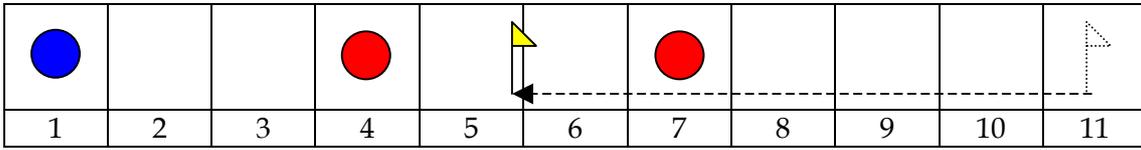


Figure 5 Effect of the Second Flag Alternative

The third alternative is given by

$$Z_{F3} = W_F (E + A)^{1-\alpha} \left(\frac{D_{F, new} - D_{F, old}}{D_{F, old}} \right) \quad (8)$$

which assumes that there are as many flags as detected entities ($E+A$). In the simple scenario there would now be multiple flags (2 in this case as there are only two other entities detected) positioned at location 11 as illustrated in Figure 6 below.

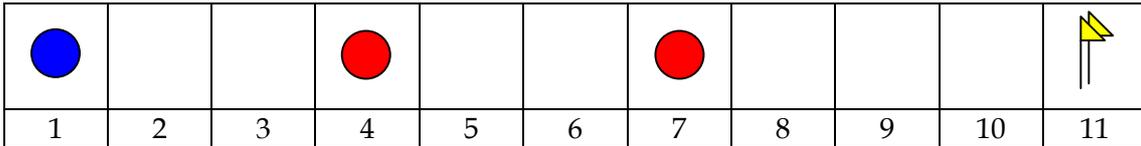


Figure 6 Effect of the Third Flag Alternative

The fourth alternative is given by

$$Z_{F4} = W_F (E + A)^{1-\alpha} \left(\frac{D_{F, new} - D_{F, old}}{\frac{1}{E} \sum_{i=1}^E D_{i, old} + \frac{1}{A} \sum_{i=1}^A D_{i, old}} \right) \quad (9)$$

which assumes that there are multiple flags and uses the scaling factor that is the average distance of all detected enemy and ally entities. In the simple scenario there

would now be multiple flags (2 in this case) in essence now located between locations five and six as illustrated in Figure 7 below.

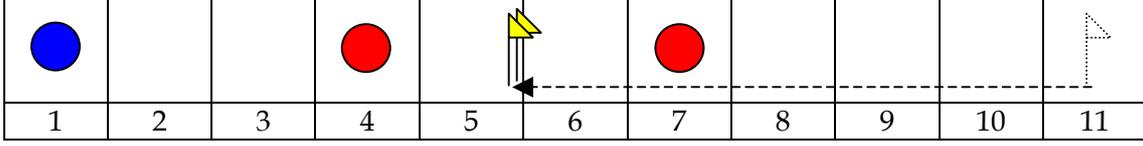


Figure 7 Effect of the Fourth Flag Alternative

Thus, in summary the proposed alternative penalty functions are given by

$$Z_{new} = \left(\frac{W_E}{E^\alpha} \sum_{i=1}^E \frac{D_{i,new} - D_{i,old}}{D_{i,old}} \right) + \left(\frac{W_A}{A^\alpha} \sum_{i=1}^A \frac{D_{i,new} - D_{i,old}}{D_{i,old}} \right) + Z_{Fj} \quad (10)$$

where Z_{Fj} is one of the flag component alternatives above ($j=1, 2,3,4$). Each of these forms appeals intuitively to the authors to what is envisaged by the attraction-repulsion weighting system. Neither could confidently be claimed as ‘correct’ and the others ‘incorrect’. We thus examine the behaviours of all alternatives in Section 3.5 below.

3.4 Spatial Interpolation Technique

Another possible algorithm, which also provides feasible behaviours, can be constructed using a spatial interpolation technique. A common problem that frequently arises in the study of spatially varying phenomena is to estimate the value of a variable at an unsampled location, or on a regular grid of unsampled locations to be used for contouring.

The objective of estimating spatially varying phenomena is to produce a representative surface that captures the spatial variation of the data and is not recognizably inconsistent with that data, and to this end a large number of techniques have been developed (see [6] for a review).

To estimate the value of a variable at an unsampled location it is intuitive to use nearby sample values and to do so in a linear fashion:

$$z(\mathbf{x}_0) \approx z^*(\mathbf{x}_0) = \sum_{i=1}^N w_i z(\mathbf{x}_i)$$

Here z^* is the estimate of the true value z at a specific location \mathbf{x}_0 ; $z(\mathbf{x}_i)$, $i = 1, \dots, N$ are N nearby sample values at locations \mathbf{x}_i , and w_i are the N weights to be chosen. The N

sample values to be used are determined by specifying some neighbourhood to be searched. The estimation problem is then to choose the weights w_i so that the distribution of the estimation errors $r(\mathbf{x}_0) = z^*(\mathbf{x}_0) - z(\mathbf{x}_0)$ is minimized.

Inverse distance estimators date back to the 1920's and are still frequently used and form part of most commercial contouring packages. These techniques give greater weight to closer samples by assigning

$$w_i = K \|\mathbf{x}_i - \mathbf{x}_0\|^{-r}, \quad i = 1, \dots, N$$

where $\|\cdot\|$ is a distance norm chosen to account for possible anisotropies, r is a non-negative parameter chosen to reflect the degree of spatial continuity, and K is a normalizing constant such that the weights sum to one for unbiasedness considerations.

To use the concepts underlying the inverse distance estimators to produce a penalty function for a movement algorithm, we replace the values $z(\mathbf{x}_i)$ with the weightings assigned to detected ally and enemy entities and the flag. Thus, within the sensor range, we 'observe' values of some variable (the penalty in this case) at a discrete set of irregularly spaced locations. Then, by applying the inverse distance estimator to approximate the variable at each of the possible new locations produces the required penalty values.

Thus, a proposed penalty function based on inverse distance estimation is given by

$$Z_{new} = \left(\frac{W_E}{E^\alpha} \sum_{i=1}^E \frac{1}{D_{i,new}^r} \right) + \left(\frac{W_A}{A^\alpha} \sum_{i=1}^A \frac{1}{D_{i,new}^r} \right) + \frac{W_F}{D_{F,new}^r} \quad (11)$$

where r is a user-specified rate factor. Note that there are two major differences in this equation to that used in inverse distance estimation. First, the normalising constant is not present. This is necessary for the situation where only the flag is visible to enable the agent to move in its direction (if assigned an appropriate weighting towards it). The second difference is the use of the factor α again in the denominators. To some extent this performs the function that the normalising constant is used for. More importantly, it is used here to provide a level of flexibility between using a cumulative and average penalty function.

As with the relative distance penalty alternatives given above, the idea is that agents that are further away are given less weight. As r tends to zero, all entities are considered to be the same distance away and the local average is used (similar to that used by MANA). As r increases the reduction in weight towards entities further away becomes more and more significant. The best move in each case would be to the location with the highest penalty (as opposed to the more usual minimization).

This new penalty function may experience similar difficulties as before with regards to the weighting to the flag. The distance to the flag is generally much larger than the distance to nearby entities and as a result the weighting to the flag may again become insignificant. Using the same thinking as before, three alternative flag components are proposed in addition to that of equation (11) above, so that we have:

$$\text{IDE } Z_{F1} = \frac{W_F}{D_{F,new}^r} \quad (\text{one flag and the actual distance to it})$$

$$\text{IDE } Z_{F2} = \frac{W_F}{D_{NF,new}^r} \quad (\text{one flag but closer than the actual flag})$$

$$\text{IDE } Z_{F3} = (E + A)^{1-\alpha} \frac{W_F}{D_{F,new}^r} \quad (\text{multiple flags and the actual distance to the flag})$$

$$\text{IDE } Z_{F4} = (E + A)^{1-\alpha} \frac{W_F}{D_{NF,new}^r} \quad (\text{multiple flags but closer than the actual flag})$$

where $D_{NF,new}$ is the distance from the potential new location of the entity to a new imaginary flag. This new flag is positioned in line with the old flag but at a distance that is equal to the average distance that all visible entities are away from the entity in question.

3.5 Simplified One-Dimensional Scenario Results

The simplified one-dimensional scenario is re-examined with each of the new penalty functions. Three values of α (0, 0.5 and 1) and r (0.1, 1 and 2) were used and the results are summarised in Table 5 below.

	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
Z_{F1}	SS	SS	SS
Z_{F2}	MS	MS	MS
Z_{F3}	SM	SS	SS
Z_{F4}	MM	MM	MS
	$r = 0.1$	$r = 1$	$r = 2$
IDE Z_{F1}	SS	SS	SS
IDE Z_{F2}	MM	MM	MM

*Table 5 New Penalty Function Results on the Simplified One-Dimensional Scenario.
Definitions: SM = Stay then Move, MS = Move then Stay, SS = Stay then Stay, MM = Move then Move*

There are four possible combinations that could occur for the two cases of sensor range of 3 and sensor range of 6, respectively. The Stay then Move combination (listed as SM) is the only one that is definitely illogical. The Move then Stay (MS) possibility is logical while the other two (SS or MM), where the entity either stays or moves regardless of the sensor range, are also both feasible depending on the interpretation of the weighing system by the user. Recall that EINSTEIN produced the SM behaviour, while MANA produced MM.

It is impossible from the results to say which scheme is the best, and this is certainly not the intent. However it is possible to state which schemes give counter-intuitive results and which ones give more display logical behaviours. The behaviour displayed by both Z_{F1} and Z_{F2} appears to be independent of the value of α and both give feasible results (SS and MS).

For the Z_{F1} case it is quite logical that if the Blue entity sees a Red entity between it and the flag that it will stay in place, and likewise if there are two Red entities. The Z_{F2} case allows the Blue entity to move if there is only one Red between it and the flag but will not move if the number of Reds increases to two. Again this is quite feasible, which one is preferred depends on how the user would want the entities to behave in the given situation.

The Z_{F3} case produced the same counter-intuitive behaviour as the EINSTEIN algorithm for the $\alpha=0$ case. However this problem was rectified for values of $\alpha=0.5$ and 1. The Z_{F4} case was generally less cautious about making a move towards the flag. This is a result of having multiple flags placed at the location of the average distance to all entities.

The results for the IDE Z_{F1} scheme seem to indicate a reluctance to move at all because of the presence of an enemy (or two) that is closer than the flag is. This may be intended by the user and therefore be perfectly legitimate. Meanwhile the results for the IDE Z_{F2} scheme show that the entity is always prepared to move forward as a result of the "closer" flag. The results for IDE Z_{F3} and Z_{F4} have been deliberately left out here because in all cases except one ($\alpha=0$ and $r=0.0001$) the results for Z_{F3} were the same as Z_{F1} . Similarly all moves for Z_{F4} were identical to Z_{F2} . Each of the different IDE flag components, except for the case already mentioned, produced results that were independent of the value of α .

3.6 Two-Dimensional Scenario Results

In addition to the simple analysis above a second set of trials were run using a 24 by 24 grid consisting of 60 enemy entities and 20 ally entities randomly distributed across the grid. For simplicity, these entities remain stationary throughout the trials. A flag was located at position (1,4) and the initial position of the entity was (24,12). The single entity whose path we wish to examine is attracted to the flag (with weighting of +100) and allies (with weighting of +20) and repelled from the enemy (with weighting of -50).

The paths for the different movement algorithms and various sensor ranges are shown in the figures below. For clarity, not all paths corresponding to all possible combinations of the various flag components and values of α are shown as many of these paths overlap or are very similar to others. Two separate graphs are shown for each sensor range for further legibility.

Figure 8 displays the paths when the sensor range is equal to 10. The results show only two distinctly different paths being taken. Using the Z_{F3} flag component (multiple flags at the actual distance) allowed the entity to reach the flag. However of all possible combinations of flag components and values of α this was the only occasion when the flag was reached. When α was 0.5, flag components Z_{F1} (one flag, actual distance) and Z_{F2} (one flag, average distance to entities) began along the same path but stopped at around (18,6).

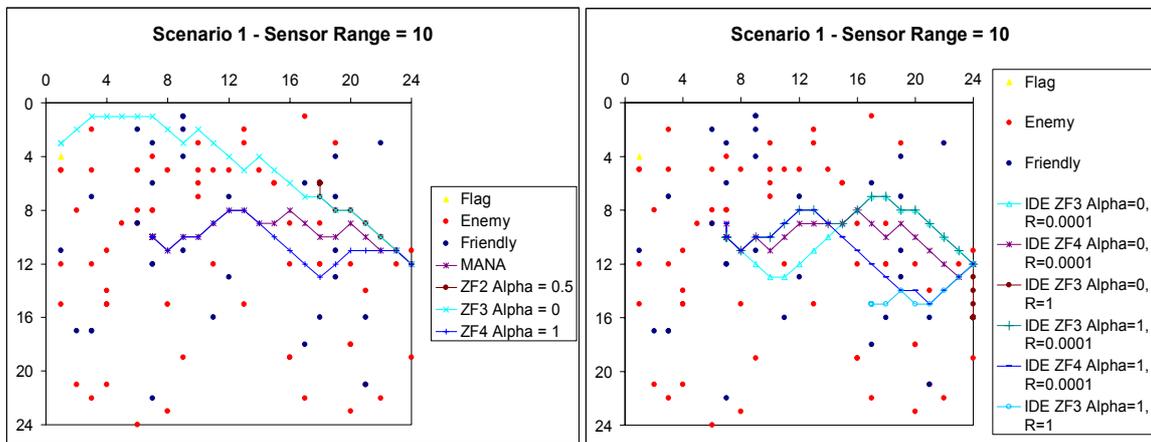


Figure 8 Variation in Movement Paths -- Sensor Range = 10

It seems apparent that these algorithms place more importance on staying close to other allies rather than pursuing the flag. However this also depends on the value of α . For the same two algorithms when α was set to zero the entity refused to move at all but when α was one, the entity followed a similar path to the dark Blue Z_{F4} path.

This last path was the most common path among the new algorithms (including those not shown above) and was similar to both the MANA and EINSTEIN paths. It appears that these algorithms move the entity to a stage where they are reluctant to move, not only because of allies around them, but also because of enemies in front of them. This was also the case for all of the IDE algorithms (Figure 8 on the right). Whilst they do not follow the exact same path, most of them ended up at the same final position that the Mana and EINSTEIN algorithms produced.

When the sensor range of the entity is decreased to five (Figure 9), two distinctly different paths are once again formed. The main difference here is that only the MANA

and EINSTEIN paths follow the upper path. They appear to get themselves into a position where they want to advance to the flag but the risk from the number of enemies surrounding them is too great. For the lower path it appears that the entity follows a certain path so as always to stay close to other allies and eventually comes to a situation again where the risk associated with moving towards the enemy is too great.

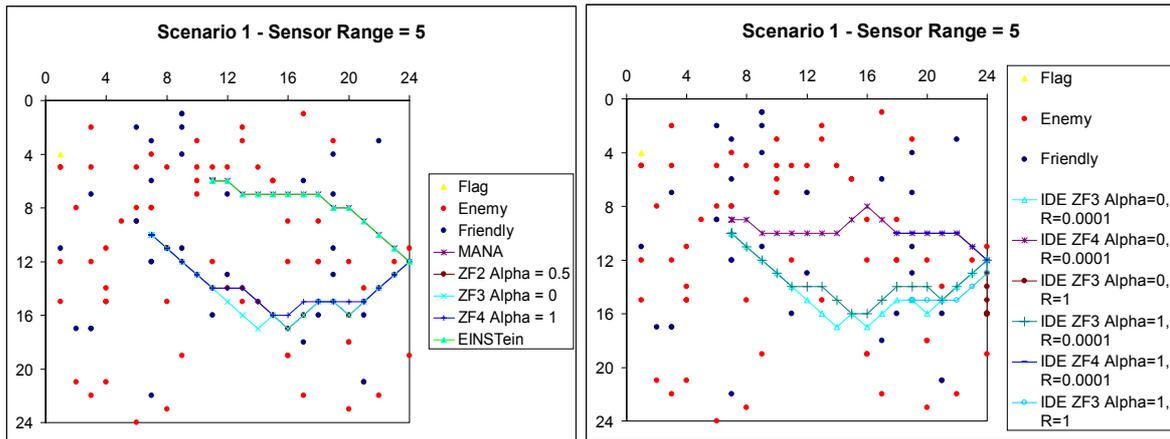


Figure 9 Variation in Movement Paths -- Sensor Range = 5

It should also be noted that all of the new algorithms (including those not shown above) produced a very similar path regardless of the value of α of the type of flag component. Intuitively it appears that the lower path would be the safer, and thus more preferable, route.

The IDE Z_{F4} algorithm produced a different path again which tended to go straight across the grid. The other IDE algorithms produced paths almost identical to the “safer” paths produced by the original Z_{F3} and Z_{F4} algorithms.

When the sensor range was decreased further (Figure 10) the same two paths as for sensor range equal to five were formed. However this time the MANA and EINSTEIN algorithms did not choose the upper path. The only two cases where the upper path was chosen were when flag components Z_{F2} and Z_{F4} were used and α was one. All other algorithms chose the lower path.

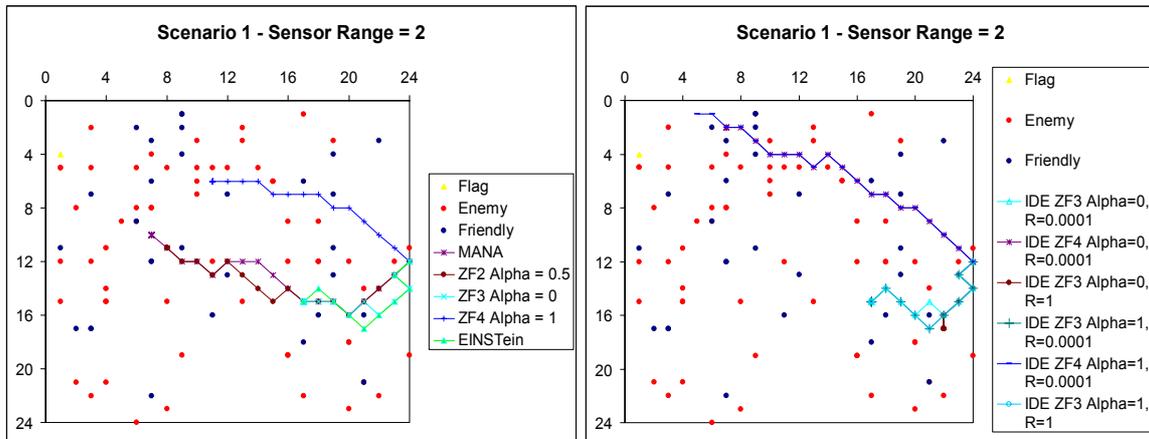


Figure 10 Variation in Movement Paths -- Sensor Range = 2

This is also the first example where the MANA and EINSTEIN paths differ. Whilst they initially follow the same general direction, the MANA path continues to the previously seen final position at (7,10) while the EINSTEIN path refuses to move any further than (17,15). All of the new algorithms also follow one of these two paths. The Z_{F1} and Z_{F3} cases always stop at (17,15) regardless of the value of α , while the Z_{F2} and Z_{F4} cases finish at, or near, (7,10) for α equal to 0 or 0.5.

The IDE ZF4 algorithms produce a path where the flag is almost reached and is similar to the ZF3 path when the sensor range was equal to ten. The other IDE algorithms produce very similar results to the EINSTEIN algorithm.

From Figures 8 -- 10, together with the complete lists of paths for all possible combinations of Z_{Fi} and $\alpha = 1, 0.5$ and 0, some comparisons can be made. In all three cases flag component Z_{F1} with α equal to zero produced little or no movement at all. This seems to suggest that using the cumulative penalty ($\alpha = 0$) requires a different flag component (higher weighting via either a closer distance or multiple flags) to allow movement. This is supported by the fact that in nearly all other cases for α equal to zero, significant movement occurred.

It is also interesting to note that penalties Z_{F3} and Z_{F4} were independent of the value of α for two out of the three sensor ranges (Z_{F3} produced a different path for $\alpha = 0$, sensor range = 10, while Z_{F4} produced a different path for $\alpha = 1$, sensor range = 2). Z_{F4} actually had an identical final position for every combination of α and sensor range except for the previously mentioned case.

In terms of comparisons with EINSTEIN and MANA it should be noted that Z_{F4} with α equal to zero or 0.5 produced a similar (within one square) final position to the MANA algorithm for all sensor ranges. The EINSTEIN algorithm was mimicked across all sensor ranges by Z_{F3} ($\alpha = 0.5$ or 1) and Z_{F1} ($\alpha = 1$). This seems logical because as α

approaches one the algorithms become more like the EINSTEIN algorithm and use the average distance rather than cumulative distances.

4. Other Considerations

4.1 Meta-Personalities

Meta-personalities are additional rules that may be used to alter the default personality of an agent depending on certain threshold conditions. They allow the agent to have some degree of adaptability to extreme situations. The three most common meta-personalities (and which both EINSTEIN and MANA implement, but with slight variations) are the cluster, advance and combat meta-personalities.

Some of the counter-intuitive behaviours that are produced by the EINSTEIN and MANA ABD's can be partially corrected by using these meta-personalities. However, there are still some limitations and artificialities associated with these meta-personalities.

4.1.1 Cluster and Advance Meta-Personalities

In both EINSTEIN and MANA the cluster meta-personality is activated when the number of allies within an agent's threshold range (EINSTEIN) or sensor range (MANA) is above a user-defined threshold. When this occurs the weighting towards other allies is temporarily set to zero. This is intended to discourage the clustering of like-coloured entities above a certain density.

The advance meta-personality is activated when the number of allies within the threshold range (EINSTEIN) or sensor range (MANA) is less than a user-defined threshold. When this occurs the weighting towards the flag (EINSTEIN) or next waypoint (MANA) is temporarily negated. This is intended to discourage the advance to the goal unless a certain level of support exists.

The first observation to make is the different means of altering the default personality weightings (either set to zero or negated). It is not clear to the authors the rationale behind this seemingly arbitrary choice. It would appear to be more consistent to either both set to zero or to both negate the weighting.

Even better perhaps would be to allow the user to choose the alternative weighting. While this increases the number of parameters the user has to specify, it does provide an increased level of flexibility to the behaviours modelled, and avoids unnecessary artificialities.

4.1.2 Combat Meta-Personality

There also appears to be an inconsistency with EINSTEIN's implementation of the combat meta-personality. This meta-personality is activated when the difference between the number of allies in the given agent's threshold range and the number of enemies in the given agent's sensor range is less than a user-defined threshold.

It is not clear to the authors why either the threshold range is not used for both counts (as with the cluster and advance EINSTEIN meta-personalities) or the sensor range is used in both counts (as with the MANA meta-personalities).

A problem can therefore occur when an agent has a large sensor range and thus possible for many enemy agents to be a long way from the agent. In this situation, the combat meta-personality could be activated and the agent not move forward since it can see a large number of enemy entities (due to its large sensor range). However, the agent doesn't take into account the fact that these enemy entities are quite far away and in fact it may be quite feasible (or safe) to move forward.

It would appear to be more consistent to use the same range (either threshold range or sensor range) when counting up the number of allies and enemies. MANA has no threshold range and uses the sensor range, however in the case of EINSTEIN it would appear more logical to use the threshold range. This would mean that entities would be more concerned about enemies that are closer than those further away.

Also, as with the advance meta-personality, the weighting towards enemies is temporarily negated under the combat meta-personality and the same comments as above apply.

It is also interesting to note that the combat rule only changes this weighting (i.e. towards enemies). It may be desirable for other personalities to be changed when an entity is outnumbered, for example they may have a stronger weighting towards allies or seek to improve their stealthiness (available only in MANA).

It would also appear more intuitive for the threshold calculation to be based on a ratio rather than a difference. Currently if the combat threshold was set at three it would imply, for example, that four allies would attack one enemy, however it would also imply that 54 allies would attack 51 enemies, which may not be desirable. If the value three was interpreted as a ratio then there would need to be at least three times more allies than enemies to attack (irrespective of the absolute number). This appears to be more in line with historical (and current) warfare practices.

4.1.3 MANA Trigger States

The comments above suggest that to remove the artificialities associated with the current implementation of meta-personalities within EINSTEIN and MANA, and to provide increased flexibility to the user, then more than a negation or a cancellation of the weightings towards enemy or the flag should be used. Indeed, substantial flexibility would tend to result if all of the default personalities, and even some of the agent's characteristics, could be temporarily modified.

Perhaps the simplest implementation of this would be via the use of the trigger states that MANA contains. Currently, MANA allows the modification of most of an agent's properties, for a user-defined period of time, on the activation of certain triggers. These triggers currently include detecting or firing at an enemy or being fired upon by the enemy. These triggers are very much action-oriented and binary in nature (detected the enemy or not). However, there is no reason why the softer triggers of cluster, advance and combat (based on counts exceeding thresholds) could not be implemented.

These meta-personality trigger states could be implemented for a period of one time-step (to reflect the current implementation within EINSTEIN and MANA), after which the agent would revert back to its default state. However, it may be preferable in certain situations for this altered behaviour to continue for some longer time frame, and the implementation via MANA trigger states certainly allows this flexibility.

4.2 Utility Curves

At the moment both EINSTEIN and MANA, and the new penalty functions proposed above, implement a simple linear utility function. That is, if we denote $X=D_{i,j}/D_{i,o}$ as the relative distance of the new location from the i -th entity to the current location from the i -th entity, then the penalty functions are based on the utility function $W*(X-1)$, where W is the weighting assigned to the i -th entity type.

This means that if the new location is such that the relative distance remains the same ($D_{i,j}=D_{i,o}$ and thus $X=1$) then the change in utility is zero. This is perfectly reasonable. However, it also implies that to obtain 'full' positive utility (in the sense of receiving all of W) then the new location must be such that the relative distance is doubled ($D_{i,j}=2*D_{i,o}$ and thus $X=2$). Similarly, it also implies that to obtain 'full' negative utility (in the sense of receiving all of $-W$) then the new location must be such that the relative distance is zero ($D_{i,j}=0$ and thus $X=0$).

It appears to the authors that there may be situations where this change in utility would not be linear. For example, it sounds equally intuitive that full positive and full negative utility be assigned if the relative distances were doubled and halved. This would require some form of non-linear utility function.

Since the exact form of this utility function may depend on the situation or scenario being modelling, it would therefore be more beneficial to allow the user to be able to define the form of the utility function for assigning the relative weights to entities based on distance.

To maintain simplicity of design, it is proposed that the user be asked to select two points (D_{\min} and D_{\max}) that represent the relative distances (relative to a distance of one which assigns zero utility) at which the full positive and full negative weight should be given. From this data, a quadratic utility function can be constructed and subsequently used in the penalty function.

It should be pointed out that a more recent ABD known as Socrates [7] does implement a non-linear utility curve, however it's movement diagnostics are not as straight forward as the simple attraction-repulsion weighting system used in EINSTEIN and MANA which has been the focus of this paper.

4.3 Stochastic Movement

The MANA model has attempted to allow some form of randomness into the movement algorithm to prevent unwanted static behaviour. This is achieved essentially to limiting the number of decimal places kept in the penalty calculation, so that proportionally more ties are encountered and a random draw is used to select the new location.

An alternative method proposed here is to interpret the (appropriately scaled) penalty at each new location as the probability of moving there. This appears to the authors to be a more natural and flexible approach, since all potential locations are candidates for the move selection but with appropriate relative frequencies.

This approach has parallels with the technique of simulated annealing [8]. In both cases, it may be desirable for poorer solutions to be (temporarily) selected, in order to avoid being trapped in a local optimum and to increase the chances of discovering a better solution on the next penalty calculation.

5. Summary

Based on counter-intuitive results from a recent study conducted by the authors using the EINSTEIN ABD, the movement algorithms of that ABD and the MANA ABD were investigated.

A simplified one-dimensional scenario was used to deduce the causes for the unwanted behaviour and some simple analysis led to a basic suggestion for

improvement to the penalty function based on relative distances. A more novel technique based on the concepts underlying spatial estimation was also proposed as an alternative.

All of these techniques were then compared on both the simplified one-dimensional scenario as well as a static two-dimensional scenario. The results show that these penalty functions do not suffer from the counter-intuitive behaviours that appear to limit the validity of the EINSTEIN and MANA penalty functions.

However, of these new penalty functions there has been no attempt to suggest which one is 'correct' or 'best', as this would be a futile exercise. The intent of this paper has been to illustrate the variability of movement paths that the alternative penalty functions can generate.

Having achieved this, we then suggest that the correct approach is to make the penalty function a user-defined 'parameter', in exactly the same way that the entity capabilities and personalities are. At a minimum, it would make more explicit the assumptions made about the movement algorithm when one provides the results or conclusions of a study. Ideally, the robustness of these conclusions should be tested to variations in the movement algorithm, in the same way that sensitivity analysis is applied to other more traditional parameters.

Some improvements to the current implementation of the meta-personalities were also made to improve their flexibility, and suggestions for extending the penalty function to incorporate non-linear utility and stochastic movement were also provided.

The authors hope that any future versions of EINSTEIN, MANA or other new ABD take into consideration the points raised in this paper.

6. References

1. A. Ilachinski, *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Combat*, Military Operations Research, Vol 5, No. 3, pp 29 - 46, 2000.
2. M.K. Lauren and R.T. Stephen, *Using Fractals to Understand the Role of Entropy in Complexity Science: An Examination of the MANA Combat Model*, Fractals (in preparation)..
3. M.K. Lauren and R.T. Stephen, *MANA Map Aware Non-uniform Automata Version 1.0 Users Manual*, NZ Defence Technology Agency, June 2001.
4. A. Ilachinski, *Enhanced ISAAC Neural Simulation Toolkit (EINSTEIN); An Artificial-Life Laboratory for Exploring Self-Organized Emergence in Land Combat, Beta-Test User's Guide*, Center for Naval Analyses, Alexandria, CIM 610.10, November 1999.

5. A.W. Gill, R. Egudo, P.J. Dortmans and D. Grieger, *Supporting the Army Capability Development Process Using Agent Based Distillations - A Case Study*, Journal of Battlefield Technology, (submitted).
6. D.F. Watson, *A Guide to the Analysis and Display of Spatial Data*, Pergamon Press, New York, 1992.
7. *Socrates v1.1 User's Manual*, Emergent Information Technologies, Inc., Vienna, VA, 2001.
8. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes, The Art of Scientific Computing (Fortran Version)*, Cambridge University Press, Cambridge, pp 326 - 334, 1989.

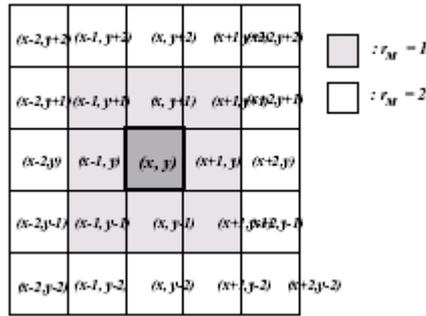
Appendix A: ISAAC Movement Algorithm

The following text is taken directly from the User Manual [4] for ISAAC Version 1.8.4 dated April 1997. This ISAAC code forms the 'engine' of the EINSTEIN code and thus the movement algorithms used by both are the same.

In the current version of ISAAC, each ISAACA can choose to move from its current position at time t -- say, (x_t, y_t) -- to any of the sites that are either a distance 1 (if the movement range is equal to $r_M=1$) or 2 (if the movement range is equal to $r_M=2$) from (x_t, y_t) ; see Figure 8. It can also select to "do nothing" and remain at its current position. Each site of the battlefield lattice may be occupied by at most one ISAACA at a given time.

An ISAACA's personality weight vector is used to rank each possible move according to a *penalty function*. The penalty function effectively measures the total distance that the ISAACA will be from other ISAACAs (which includes both friendly and enemy ISAACAs) and from its own and enemy flags, each weighted according to the appropriate component of the personality weight vector, w . An ISAACA moves to the position that incurs the least penalty, or the move that best satisfies the ISAACA's personality-driven desire to "move closer to" other ISAACA's in given states and either of the two flags.

Figure 8. Set of possible ISAACA moves from its current (x, y) position



The general form of the penalty function is given by:

$$\begin{aligned}
 Z(x, y) = & w_1 s_{\text{red}}^{-1} N_{\text{alive red}}^{-1} \sum_{\text{alive red}; i} d[i; (x, y)] + \\
 & w_2 s_{\text{blue}}^{-1} N_{\text{alive blue}}^{-1} \sum_{\text{alive blue}; i} d[i; (x, y)] + \\
 & w_3 s_{\text{red}}^{-1} N_{\text{injured red}}^{-1} \sum_{\text{injured red}; i} d[i; (x, y)] + \\
 & w_4 s_{\text{blue}}^{-1} N_{\text{injured blue}}^{-1} \sum_{\text{injured blue}; i} d[i; (x, y)] + \\
 & w_5 d_{\text{new}}[\text{red flag}; (x, y)] / d_{\text{old}}[\text{red flag}; (x, y)] + \\
 & w_6 d_{\text{new}}[\text{blue flag}; (x, y)] / d_{\text{old}}[\text{blue flag}; (x, y)] .
 \end{aligned}$$

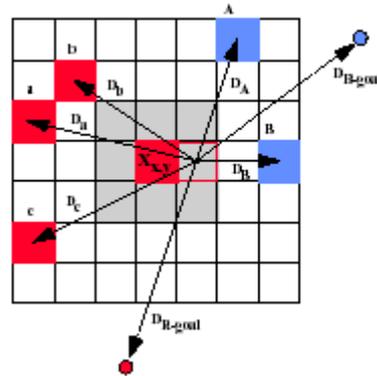
where w_i are the components of the personality weight vector and are Red and $s_{red} = \sqrt{2} r_{red}$, $s_{blue} = \sqrt{2} r_{blue}$, S_{blue} scale factors, respectively, $d[i;(x,y)]$ is the distance between the i -th element of a given sum and the ISAACA positioned at (x,y) , N_i is the total number of elements within the given ISAACA's sensor range, and d_{new} and d_{old} represent distances computed using the given ISAACA's new (i.e., candidate move) position and old (i.e., current) position, respectively. For example, the first summation appearing at the top of the above expression represents the sum of distances from the position (x,y) to all Red alive ISAACAs located within the sensor range box of position (x,y) . In the case of a Red ISAACA, this sensor range box is defined by sensor range $r_{red,S}$; in the case of a Blue ISAACA, it is defined by sensor range $r_{blue,S}$.

A "penalty" is computed for each possible move: Z_1, Z_2, \dots, Z_N . If the movement range $r_M=1$, $N=9$; if $r_M=2$, $N=25$. The actual move is the one that incurs the least penalty. If there is a set of moves (consisting of more than one possible move) that incur exactly the same minimum penalty, an ISAAC randomly selects the actual move from among the candidate moves making up that set.

Example

Figure 10 shows a portion of the notional battlefield surrounding a Red ISAACA X positioned at (x,y) . There are three Red ISAACAs (a, b and c at distances D_a, D_b and D_c from X , respectively) and two Blue ISAACAs (A and B , at distances D_A and D_B from X , respectively) within X 's sensor range.

Figure 10. Sample penalty calculation



Assuming that X 's movement range $r_M=1$, X 's next move is determined by minimizing the penalty $Z(x',y')$ that will be incurred by selecting each of the nine nearest neighboring sites, and $(x' = x, y' = y)$ and $(x' = x \pm 1, y' = y \pm 1)$ (shown in gray in Figure 10). The penalty is given explicitly by

$$Z(x',y') = w_1 s_{red}^{-1} \left(\frac{1}{3} \right) [D_a + D_b + D_c] + w_2 s_{red}^{-1} \left(\frac{1}{2} \right) [D_A + D_B] + w_5 \left(\frac{D_{B-goal}}{D_{B-goal}^0} \right) + w_6 \left(\frac{D_{R-goal}}{D_{R-goal}^0} \right),$$

where $D_{R\text{-goal}}$ and $D_{0\ R\text{-goal}}$ are the distances from (x,y) and (x',y') to the Red goal, respectively, and $D_{B\text{-goal}}$ and $D_{0\ B\text{-goal}}$ are the distances from (x,y) and (x',y') to the Blue goal, respectively.

Appendix B: MANA Movement Algorithm

The following text is taken directly from the User Manual [3] for MANA Version 1.0 dated June 2001.

The most important action of an agent is to move. The movement algorithm selects the grid square within its movement range that most satisfies its desire to move towards some entities and away from others. The current location is also considered, so the agent can stay put.

The movement algorithm consists of the following steps:

- Consider all moves within the movement range of the agent, including staying put.
- Eliminate moves into locations containing other agents or impassable terrain.
- Considering all the entities in range: decide on the most appealing of the permissible moves, using personality weights that represent a desire to move toward or away from agents, the waypoints, terrain or contacts on the Situational Awareness map.
- Impose behaviour modifiers that change the basic behaviour (e.g. minimum distance to others, cluster constraints, and so on). If a number of moves are nearly equal, then choose a move at random from the attractive moves.

Local sensor information takes precedence over information available to the agent from the situational awareness map. If an enemy is within sensor range, then the influence of the situational awareness is ignored.

The penalty calculation finds the move with the least “penalty”. Moves are possible to grid squares within “movement speed” squares of the current location and not already occupied by an agent or impassable terrain.

If several moves have a similarly low penalty, a move is chosen at random from the good moves. The “movement precision” parameter sets how wide the margin should be for accepting similarly good moves. Setting the movement precision to a low value will mean that most often only the best move will be chosen and the movement will appear very deterministic. If the movement precision is too great, the agents tend to wander about in a Brownian motion, as moves are selected at random. See section 3.3.1 in [3] for a description of how to use the precision parameter.

The tendency to move toward or away from an entity is constant with distance. For example, the weighting to move towards the next waypoint is the same whether the entity is three cells or 150 cells away from it. The penalty for moving to any grid location is the sum of 10 penalty calculations, corresponding to the 10 personality parameters listed in Table 1.

The algorithm used to calculate the penalty for a collection of entities within sensor range is the same for all 10 components. The general algorithm for calculating the penalty component associated with a candidate move is shown in Figure 9. The important term in the algorithm is: $(NewDist+(100-OldDist))/100$.

This treats all entities as if they are about 100 units away. If $NewDist < OldDist$ and a move closer is desirable, the penalty term will end up slightly less than 1.0. If $NewDist > OldDist$ and a move away from the entity is desirable, a penalty term of slightly greater than 1.0 will result i.e. greater penalty.

```
Given a candidate move...
Loop through all entities within range of agent's current location
  OldDist = units from current location to entity
  NewDist = units from candidate location to entity
  If NewDist < Minimum then Direction = -1 else Direction = 1
  Sum = Sum + Direction * (NewDist+(100-OldDist))/100
  NumEntities = NumEntities + 1
Repeat loop
Penalty = Sum / NumEntities
```

Figure 9: Penalty Calculation Algorithm

The penalty for moving towards or away from other agents is normalized by the number of agents (the last line of Figure 9). For example, if I am an agent attracted to friendly agents, I seek to minimize my *average* distance to friendly agents within my sensor range. However, a number of constraints can modify this attraction. The minimum distance to friends/enemies range sets a radius to other agents, inside which the penalty is negated. The check against minimum distance is made for every agent within sensor range. If an agent is within the minimum distance, its penalty is negated before adding to the sum for the friends penalty component.

DISTRIBUTION LIST

Title

Author(s)

AUSTRALIA

Copy No.

DEFENCE ORGANISATION

Task Sponsor

S&T Program

Chief Defence Scientist
FAS Science Policy
AS Science Corporate Management
Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser Policy and Command
Navy Scientific Adviser (Doc Data Sheet and distribution list only)
Scientific Adviser - Army (Doc Data Sheet and distribution list only)
Air Force Scientific Adviser
Director Trials

} shared copy

Aeronautical and Maritime Research Laboratory

Director

Electronics and Surveillance Research Laboratory

Director (Doc Data Sheet and Distribution List only)

Chief of Division

Research Leader

Head (where appropriate)

Task Manager

Author(s):

DSTO Library

Library Fishermans Bend

Library Maribyrnong (Doc Data Sheet only)

Library Salisbury (2 copies)

Australian Archives

Library, MOD, Pyrmont (Doc Data sheet only)

Library, MOD, HMAS Stirling

US Defense Technical Information Center, 2 copies

UK Defence Research Information Centre, 2 copies

Canada Defence Scientific Information Service, 1 copy

NZ Defence Information Centre, 1 copy

Capability Systems Staff

Director General Maritime Development (Doc Data Sheet only)
Director General Land Development
Director General Aerospace Development (Doc Data Sheet only)

Knowledge Staff

Director General Command, Control, Communications and Computers (DGC4)
(Doc Data Sheet only)
Director General Intelligence, Surveillance, Reconnaissance, and Electronic Warfare (DGISREW)R1-3-A142 CANBERRA ACT 2600 (Doc Data Sheet only)
Director General Defence Knowledge Improvement Team (DGDKNIT)
R1-5-A165, CANBERRA ACT 2600 (Doc Data Sheet only)

Navy

SO (SCIENCE), COMAUSNAVSURFGRP, BLD 95, Garden Island, Locked Bag 12, PYRMONT NSW 2009 (Doc Data Sheet and distribution list only)

Army

SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), MILPO Gallipoli Barracks, Enoggera QLD 4052 (Doc Data Sheet only)

Air Force**Intelligence Program**

DGSTA Defence Intelligence Organisation
Manager, Information Centre, Defence Intelligence Organisation

Acquisitions Program**Corporate Support Program**

Library Manager, DLS-Canberra

SPARES (5 copies)

Total number of copies: