A Connectionist-Symbolic Approach to Modeling Agent Behavior: Neural Networks Grouped by Contexts

Amy E. Henninger[1,2], Avelino J. Gonzalez[1], Michael Georgiopoulos[1], Ronald F. DeMara[1]

[1] Intelligent Systems Laboratory
School of Electrical Engineering and Computer Science,
University of Central Florida
Orlando, FL 32816
{amy, ajg, mge, rfd}@isl.ucf.edu
[2] Soar Technology, Inc.
317 North First Street
Ann Arbor, MI. 48103-3301
amy@soartech.com

Abstract. A recent report by the National Research Council (NRC) declares neural networks "hold the most promise for providing powerful learning models". While some researchers have experimented with using neural networks to model battlefield behavior for Computer Generated Forces (CGF) systems used in distributed simulations, the NRC report indicates that further research is needed to develop a hybrid system that will integrate the newer neural network technology into the current rule-based paradigms. This paper supports this solicitation by examining the use of a context structure to modularly organize the application of neural networks to a low-level Semi-Automated Forces (SAF) reactive task. Specifically, it reports on the development of a neural network movement model and illustrates how its performance is improved through the use of the modular context paradigm. Further, this paper introduces the theory behind the neural networks' architecture and training algorithms as well as the specifics of how the networks were developed for this investigation. Lastly, it illustrates how the networks were integrated with SAF software, defines the networks' performance measures, presents the results of the scenarios considered in this investigation, and offers directions for future work.

1 Introduction

The combination of computer simulation and networking technologies has provided military forces with an effective means of training through the use of Distributed Interactive Simulation (DIS). DIS is an architecture for building large-scale simulation models from a set of independent simulator nodes that represent entities in the simulation [1]. These simulator nodes individually simulate the activities of one or more entities in the simulation and report their attributes and actions of interest to other simulator nodes via the network. DIS nodes simulating combat vehicles, such as M1 Abrams tanks, are crewed by soldiers being trained. The trainees operate the controls of the simulators as they would in the actual vehicles, and the simulators

implement actions in the simulated battlefield. Since, in a synthetic battlefield, the trainees need opposing forces against which to train, a type of DIS node known as a Computer Generated Force (CGF) system was developed.

CGFs are computer-controlled behavioral models of combatants used to serve as opponents against whom trainees can fight or as friendly forces with which the trainees can fight. At a minimum, the behavior generated should be feasible and doctrinally correct. For example, behaviors should be able to emulate the use of formations in orders, identify and occupy a variety of tactical positions (e.g., fighting positions, hull down positions, turret down positions, etc), and plan reasonable routes.

Researchers in [2], [3], and [4] have experimented with using neural networks to model battlefield behavior for CGF systems used in military simulations. This technology has been identified as one that "holds the most promise for providing powerful learning models" in a recent National Research Council Report [5]. Also asserted in this report, however, is the need for further research to develop hybrid systems that will integrate the newer neural network technology into the current rule-based paradigms. This investigation considers one such approach by using a framework based on modular decomposition to develop and apply the neural networks generating SAF behavior. Specifically, this research examines the performance improvements made to a neural network based near-term movement model by adopting a modular approach that groups neural networks according to contexts..

2 Modular Decomposition

The use of a modular approach to a modeling task can be beneficial in a variety of ways. For example, it can be used for the purposes of improving performance. In other words, although the task could be solved with a monolithic set, better performance is achieved when it is broken down into a number of expert modules. Once the task is decomposed it is possible to switch to the most appropriate module, depending on the current circumstances or context. Switching has been discussed in the control literature [6][7], as well as the literature on behavior-based robotics [8].

In addition to performance improvement, other motivations for adopting a modular approach to a problem include a reduction in model complexity and construction of the overall system such that it is easier to understand, modify, and extend. Thus the "divide and conquer" principle is used to reduce the complexity of a single net system. This enables the use of different neural net architectures or algorithms to be applied to individual sub-problems, making it possible to exploit specialist capabilities. Moreover, where appropriate, some of these components could make use of non-neural computing techniques. This justification has been noted [9][10] and is common to engineering design in general. Another motivation for adopting a modular approach is the reduction of network training times [11]. Finally, in well-defined domains, the use of a priori knowledge can be used to suggest an appropriate decomposition of a task. This approach complements the knowledge acquisition efforts

and knowledge representation paradigms used in current SAF systems [12] and can be easily extended to the acquisition of knowledge and tactics for SAF systems [13].

The decomposition of a problem into modular components may be accomplished automatically or explicitly. When the decomposition of the task into modules is determined explicitly, this usually relies on a strong understanding of the problem. The division into sub-tasks is known prior to training [14], and improved learning and performance can result. An alternative approach is one in which the task is automatically decomposed according to the blind application of a data partitioning technique. Automatic decomposition is typically applied with the intent of performance improvement, whereas explicit decomposition could have the aim of either improving performance or accomplishing tasks that might not be accomplished as easily or as naturally with a monolithic net.

3 Methodology

The synthetic force system used for the prototype development work was ModSAF, a training and research system developed by the Army's Simulation, Training, and Instrumentation Command (STRICOM). ModSAF provides a set of software modules for constructing computer-generated force behaviors at the company level and below. Typically, ModSAF models are employed to represent individual soldiers or vehicles and their coordination into orderly-moving squads and platoons, but their tactical actions as units are planned and executed by a human controller. The human behaviors represented in ModSAF include move, shoot, sense, communicate, tactics, and situation awareness. The authoritative sources of these behaviors are subject matter experts and doctrine provided by the Army Training and Doctrine Command (TRADOC). ModSAF uses finite state machines (FSMs) to represent the behavior and functionality of a process for a pre-defined number of states.

Figure 1 illustrates the scope of a Road March task through a possible representation of its FSM transition formalism. Inherent in this representation, is a temporal logic or sequencing to the state transitions in the formalism. For example, a tank would never reach an "end of road march" state (where it would slow down) before it would reach a "start of road march" state (where it would speed up). The near-term movement models addressed in this research pertain to the "Follow Route" state of the Road March FSM shown in Figure 1. In other words, the model is developed for and evaluated when the M1A2 is in a "Follow Route" state that is not influenced by proximity to the start or the end of the route. This simplifies the modeling task since the model does not have to learn to speed up and slow down at the beginning and end of the route, respectively. Further, it indicates that long-term route planning is not a part of the model's functionality. Lastly, it constrains the model's range of operability to those scenarios where there are no "Halt" states embedded in the Road March behavior.

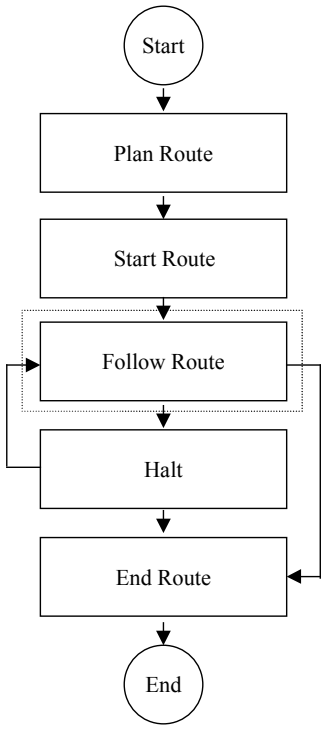

Fig. 1. FSM Representation of Road March Behavior

Figure 2 further illustrates the model addressed in this research and presents it relative to other models that might be required in a "Follow Route" state of the Road March FSM. These models blend low-level decisions with motor skills and environmental feedback. This model is responsible for the physical movement of the tank through the virtual battlefield and is represented by the change in the tank's speed and orientation.

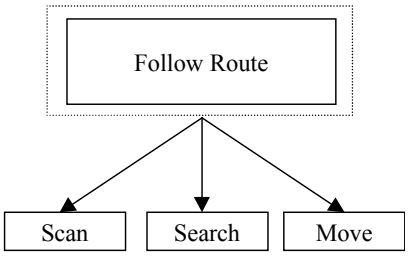

Fig. 2. Skill Models Supporting Follow-Route State

The route used for development and testing in the prototype work (see Figure 3) can be found in the section of terrain east of Barstow Road and west of Hill 720 in the

NTC-0101 terrain database. In general, the route is represented by 45 route points and is approximately 7 kilometers long. It takes the M1A2 tank about 15 minutes of simulation time to travel at 8m/s. From this route, two similarly constructed segments were selected for detailed work. Each of these segments consisted of 4 route points centered about a turn. These are labeled as segments 1 and 2 in Figure 3 and they were used for purposes of model training and testing.

Eleven scenarios were constructed for purposes for training and testing. In each of these scenarios a single ModSAF M1A2 entity was placed at X,Y position 22579,24328 with an initial heading of 359° and each entity was assigned to perform a Road March with the route boldfaced in Figure 3. These scenarios were created with identical user-supplied parameters.

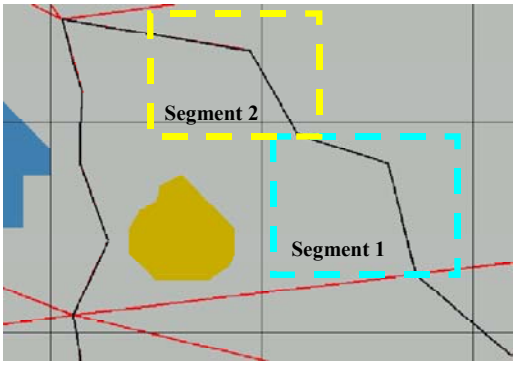

Fig. 3. Segments Used for Model Training Testing

As communicated in Table 1, a preliminary analysis of these graphs resulted in one possible partitioning of the data. These data groupings are considered with the intent to improve model generalization and system performance.

Table 1. Classification of Data According to Approach Type

Scenario Number	Approach Category
DEMO	Nominal
1	Nominal
2	Early
3	Late
4	Early
5	Late
6	Nominal
7	Early
8	Double
9	Nominal
10	Late

where “Approach Category” categorizes the approach types to a curve by distance away from the target waypoint. Specifically, this is defined by:

Early	> 45m to target waypoint
Nominal	45< > 30 m to target waypoint
Late	< 30 m to target waypoint

This categorization can be expressed as a rule-set that represents an explicit decomposition of the task. We refer to the product of this explicit task decomposition as a "context" [15]. The application of this rule results in the generation of six networks for three categories including two networks each. These two networks represent the change in the M1A2 entity's speed and the change in the M1A2 entity's heading.

All of the networks used a feed-forward architecture and were trained with back-propagation according to the delta learning rule using a momentum factor to speed the descent along the error surface. All networks were trained with randomly generated initial weights according to four different random seeds. Also, all networks used 0.01 for the training rate, η, and 0.9 for the initial momentum parameter, α. The momentum parameter was periodically adjusted to speed the rate of descent along the error surface.

Each network used a sigmoid function at the hidden nodes and a linear transformation at the output nodes. Each of these networks had 7 inputs, 20 nodes in the first hidden layer, 5 nodes in the second hidden layer, and a single output. The inputs were derived and normalized according to equations 1 – 18 below. Fundamentally, the inputs for each of the networks were a function of the M1A2 entity's state at the last simulation clock and how this state related to the road characteristics and March Order parameters.

$$S_t = S_{t-1} + \Delta S_t$$

$$\text{where } \Delta S_t = f(Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}, Hz_{t-1}) \quad (1)$$

$$\theta_t = \theta_{t-1} + \Delta\theta_t$$

$$\text{where } \Delta\theta_t = f(Ra_{t-1}, Rb_{t-1}, Rc_{t-1}, Rp_{t-1}, Rs_{t-1}, HRab_{t-1}, HRbc_{t-1}) \quad (2)$$

where

$$Ra_t = S_t / (Da_t + M) \quad (3)$$

$$Rb_t = S_t / (Db_t + M) \quad (4)$$

$$Rc_t = S_t / (Dc_t + M) \quad (5)$$

$$Rp_t = S_t P_t / M \quad (6)$$

$$Rs_t = S_t / M \quad (7)$$

$$HRab_t = Hab_{r.} \times Hxy_t \quad (8)$$

$$HRbc_t = Hbc_{r.} \times Hxy_t \quad (9)$$

$$S_t = entity\ speed\ at\ t \quad (10)$$

$$Da_t = distance\ to\ previous\ waypoint \quad (11)$$

$$Db_t = distance\ to\ current\ waypoint \quad (12)$$

$$Dc_t = distance to\ next\ waypoint \quad (13)$$

$$M = march\ order\ speed \quad (14)$$

$$P_t = perpindicular\ distance\ to\ road \quad (15)$$

$$Hab_t = direction\ of\ road\ segment\ ab \quad (16)$$

$$Hbc_t = direction\ of\ road\ segment\ bc \quad (17)$$

$$Hxy_t = entity\ orientation \quad (18)$$

As a point of comparison, this scheme of models (i.e., 2 models for each of three categories) was compared with a non-categorized set of the same training data. That is, the experimental baseline model used all of the same model parameters but did not apply the classification rule to partition the training data or control the model execution. As such, the experimental baseline model was trained with roughly 3 times more data than were any of the models in the categorized approach. Thus, by comparing the two methods (i.e., context approach according to rules and monolithic approach), the utility of the context approach can be evaluated for this application.

4 Experimental Results

Essential to the task of determining whether one model out-performed another is a metric to make such a comparison. Validating SAF models has typically been performed subjectively by SMEs and the DIS community has no known quantitative performance measure to evaluate the performance of a SAF near-term movement model. Given the level of resolution of SAF maps, it is impractical to assume that a SME could detect a noticeable difference in models due to the addition of a context shift. In other words, even if a human observer could visibly discriminate between two different types of movement models, it is unlikely that he could visibly detect the difference in the same movement model represented by a monolithic neural network versus represented by a module of networks. Because using SME validation to compare the models in this research was susceptible to error, the investigators made use of the DIS entity state synchronization concept to evaluate model performance. This was accomplished by implementing each of the models as DIS dead-reckoning models [16] and then comparing the numbers of ESPDUs generated by each of models 1 and 2. DIS dead-reckoning is a predictive contract of vehicle movement that can be used to reduce network traffic in a distributed simulation. An ESPDU is the protocol data unit used to communicate that an error between the entity's synchronization model and the entity's true position exceeds some threshold value. By communicating the vehicle's location, velocity and acceleration to other DIS simulators, the dead-reckoning models residing on these simulators can predict the physical location of the vehicle. Implicit in this measure of performance is the assumption that the model with the lower PDU count is the model that best fits the source data used to develop the model and from which the PDU count is derived.

As shown in Figure 4, the comparison of the entity's true position and the position according to the dead-reckoning model occurs in the ModSAF *libentity* library. As

such, the neural network models used in this investigation replaced the dead-reckoning code in the *libentity* library

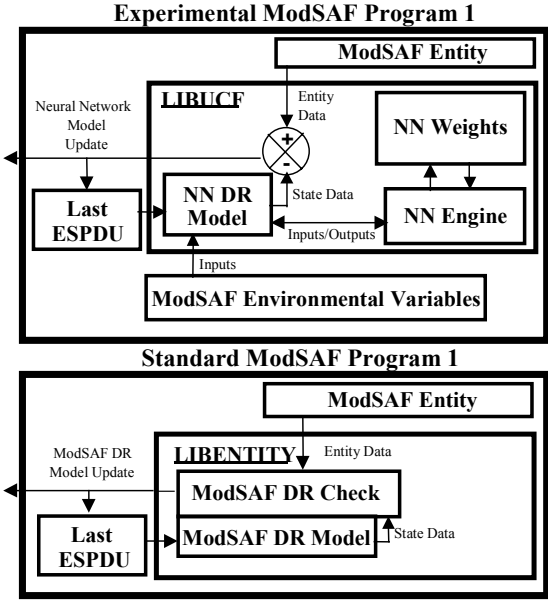

Fig. 4. Functional Relationship of Neural Networks to ModSAF Dead-Reckoning Code

Evaluating the context-based neural network scheme over segment 1 results in the PDU counts indicated in Table 2.

Table 2. Combined Segment 1 Results of Networks Trained According to Approach Type Classification Scheme

	Scenario by Classification of Segment 1 Approach										
Training on Segment1	**D N**	**1 N**	**2 E**	**3 L**	**4 E**	**5 L**	**6 N**	**7 E**	**8 D**	**9 N**	**10 L**
DEMO&1(N)	24	24	45	47	38	41	33	42	56	41	42
2 & 4 (E)	44	42	23	39	19	39	38	27	54	58	40
3 & 5 (L)	52	45	49	18	36	16	46	42	44	68	17
DIS	40	42	41	31	33	35	42	38	41	45	34
% Reduction	40	43	44	42	43	55	22	29	X	9	50

where N – nominal, E – early, L – late, D - double

From these results, it is apparent that the same category of networks consistently performs better on like categories of testing. Thus, it appears that the use of a hybrid approach is of some benefit to this problem. Using this approach would yield an

average ESPDU reduction of 45% over Scenarios involved in training (D-5) and 28% over the same category scenarios not used in training (6-10).

Evaluating this same combination of networks over segment 2 results in the PDU counts communicated in Table 3.

Table 3. Combined Segment 2 Results of Networks Trained According to Approach Type Classification Scheme

	Scenario by Classification of Segment 2 Approach										
Training on Segment1	**D N**	**1 N**	**2 E**	**3 L**	**4 E**	**5 L**	**6 N**	**7 E**	**8 D**	**9 N**	**10 L**
DEMO&1(N)	91	95	66	101	110	102	105	121	105	108	101
2 & 4 (E)	53	53	47	50	70	50	46	56	45	46	49
3 & 5 (L)	42	43	36	43	72	44	45	51	48	42	44
DIS	42	43	47	32	58	30	43	43	40	38	39

While these models do not consistently reduce PDU counts across all of the Scenarios over Segment 2, a pattern of lower PDU counts in Scenarios whose approach classification correlates with the network classification is apparent. For example, the Scenarios in Segment 2 classified as "L" yield consistently lower PDU counts with the network trained by Segment 1 Scenarios 3 and 5 (also both classified as late). Also, the Scenario in Segment 2 classified as "E" yields a lower PDU count when tested with the network trained by Segment 1 Scenarios 2 and 4 (also both classified as early). So, while the total PDU count for Segment 2 is not reduced through this modeling scheme, there does appear to be a correlation between the type of approach represented by the trained model and the approach classification of the tested segment.

As a means of comparison, the data from the scenarios used to generate the networks for the previously presented modular configuration were used to develop a single, monolithic neural network with the same parameters. That is, the entire, aggregated data set in a single network category (as opposed to three, individual data sets partitioned by Approach Type) was considered. Thus, the results of this network can be compared with the results of previously defined to determine the effects of modularizing the data on the system's performance. The results of this experiment may be seen in Tables 4 and 5, representing the evaluation of the model over Segment 1 and Segment 2, respectively.

Table 4. Combined Segment 1 Results of Monolithic Networks Trained with Data from all Scenarios DEMO,1,2,3,4,&5

	Scenario by Classification of Segment 1 Approach										
Training on Segment1	**D N**	**1 N**	**2 E**	**3 L**	**4 E**	**5 L**	**6 N**	**7 E**	**8 D**	**9 N**	**10 L**
All (D-5)	32	34	34	32	25	29	36	29	48	55	31
DIS	40	42	41	31	33	35	42	38	41	45	34

The results of this experiment indicate a reduction in ESPDUs on Segment 1 (see Table 4) and seem to suggest an ability to generalize to scenarios not used in training. However, by comparing results in Tables 2 and 4, it is apparent that neither the scenarios used in training nor the scenarios held out from the context-based network scheme outperform the current monolithic network scheme. Correspondingly, as evidenced in a comparison of Tables 3 and 5, when evaluated on Segment 2, the monolithic network scheme did not perform as well as the modularized scheme.

Table 5. Combined Segment 2 Results of Monolithic Networks Trained with Data from all Scenarios DEMO,1,2,3,4,&5

	Scenario by Classification of Segment 2 Approach										
Training on Segment1	**D N**	**1 N**	**2 E**	**3 L**	**4 E**	**5 L**	**6 N**	**7 E**	**8 D**	**9 N**	**10 L**
All (D-5)	48	47	41	52	71	52	48	55	48	48	52
DIS	42	43	47	32	58	30	43	43	40	38	30

5 Summary

In summary, a context-based neural-network modeling scheme was empirically developed and then tested in a simulated environment. As part of this effort, the use of explicit model decomposition schemes was considered as a mechanism for improved model performance. The performance of the best modeling combination was evaluated in three ways. First, the models were tested on movement methods that were used in training. Second, the models were tested on movement methods that were not used in training, but similar to those used in training. Third, the models were tested on movement methods on an entirely new part of the route other than that used in training. The modeling scheme in the first case gave an average ESPDU reduction of approximately 45% over current DIS dead-reckoning methods. The same modeling scheme in the second case resulted in an average ESPDU reduction of approximately 28% over current DIS dead-reckoning methods. Lastly, the modeling scheme in the third cases did not result in ESPDU reduction over DIS dead-reckoning methods, but did yield results consistent with the expectations of the classification scheme. These results suggests that the performance of neural networks applied to a low-level SAF reactive-task is improved by the use of the context-based task decomposition scheme. Future work in this aspect of the study includes investigating methods of automating the learning of the task decomposition and hence, the context-shifting rules. Also, the improvement of the neural networks' performance continues to be explored. This includes considering alternative types of architectures, inputs, normalization schemes, and sampling strategies. Since the ModSAF infrastructure to collect data and evaluate models is now in place, more work can be done to improve these preliminary results.

References

1. Smith, S., and Petty, M. (1992). Controlling Autonomous Behavior in Real-Time Simulation. In Proceedings of the Second Conference in Computer Generated Forces and Behavior Representation. Orlando, FL., May, 1992.

2. Crowe, M. (1990). The Application of Artificial Neural Systems to the Training of Air Combat Decision-Making Skills. Proceedings of the 12th Interservice/Industry Training Systems Conference, Orlando, FL. Nov., 1990.

3. Jaszlics, S.L. (1993). Artificial Intelligence in Tactical Command and Control Applications: Architecture and Tools. In Proceedings of the Third Conference on Computer Generated Forces and Behavior Representation. Orlando, FL., March, 1993.

4. Morrison, J.D. (1996). Real-time Learning of Doctrine and Tactics Using Neural Networks and Combat Simulations. Military Operations Research, vol. 2, no. 3, pages 45-60.

5. Pew, R.W., and Mavor, A.S., eds. (1998). Modeling Human and Organizational Behavior: Application to Military Simulations. Washington, DC: National Academy Press.

6. Murray-Smith, R., and Johansen, T.A. (1997). Multiple Model Approaches to Modelling and Control. Taylor and Francis, UK.

7. Narendra, K. S., Balakrishnan, J., and Ciliz, K. (1995). Adaptation and Learning Using Multiple Models, Switching and Tuning. IEEE Control Systems Magazine, June, 1995, pp. 37-51.

8. Brooks, R.A. (1986). A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, RA-2:14-23.

9. Gallinari, P. (1995). Modular Neural Net Systems: Training Of. In M.A., Arbib, editor, The Handbook of Brain Theory and Neural Networks, pp. 582-585. Bradford Books: MIT Press.

10. Hrycej, T. (1992). Modular Learning in Neural Networks. John Wiley, Chichester.

11. Pratt, L.Y., Mostow, J., and Kamm, C.A. (1991). Direct Transfer of Learned Information Among Neural Networks. In Proceedings of the Ninth National Conference on Artificial Intelligence (AAI-91), pp. 584-589. Anaheim, CA, 1991.

12. Ourston, D., Blanchard, D., Chandler, E., and Loh, E., (1995). From CIS to Software, In Proceedings of the Fifth Conference on Computer Generated Forces and Behavior Representation. Orlando, FL., May, 1995, pp. 275-285.

13. Henninger, A., and Gonzalez, A. (1997). Automated Acquisition Tool for Tactical Knowledge, In Proceedings of the 10th Annual International Florida Artificial Intelligence Research Symposium, Melbourne FL., May, 1997, pp. 307-311.

14. Hampshire, J.B., and Waibel, A.H. (1992). The Meta-P,I Network: Building Distributed Representations for Robust Multiource Pattern Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(7): 751-769, 1992.

15. Gonzalez, A., and Ahlers, R. (1995). Context-based Representation of Intelligent Behavior in Simulated Opponents. In Proceedings of the Fifth Conference on Computer Generated Forces and Behavior Representation, Orlando, FL., May, 1995.

16. Lin, K., and Ng, H. (1993). "Coordinate Transformations in Distributed Interactive Simulation (DIS)". Simulation, vol. 61, No. 5, Nov, 1993, pp. 326-331.