

SURVIVABILITY OF COMPLEX SYSTEM – SUPPORT VECTOR MACHINE BASED APPROACH

**Y, HONG, N. GAUTAM, S. R. T. KUMARA, A. SURANA, H. GUPTA,
S. LEE, V. NARAYANAN, H. THADAKAMALLA**

The Dept. of Industrial Engineering, The Pennsylvania State University,
University Park, PA 16802

M.BRINN

BBN Technologies, Cambridge, MA

M. GREAVES

DARPA IXO/JLTO, Arlington, VA 22203

ABSTRACT

Logistic systems which are inherently distributed, in general can be classified as complex systems. Survivability of these systems under varying environment conditions is of paramount importance. Different environmental conditions in which the logistic system resides are translated into several stresses. These in turn will manifest as internal stresses. Logistic systems can be modeled as a collection of software agents. Each agent's behavior is a result of the stresses imposed. Predicting the agents' collective behavior is of paramount importance to ensure survivability. Analytical modeling of such systems becomes very difficult, albeit impossible. In this paper, we study a supply chain in which a real life scenario is used. We implement the supply chain in Cougaar (Cognitive Agent Architecture developed by DARPA) and develop a predictor, based on Support Vector Machine. We report our methodology and results with real-life experiments and stress scenarios.

INTRODUCTION

Logistic systems can be classified as complex systems (Choi et al., 2001, Baranger, <http://necsi.org/projects/baranger/cce.pdf>). Logistic systems have many components such as suppliers and distributors at several stages. These components are distributed geographically but interdependent. At each component some form of nonlinear decision making process goes on. Typically the system would respond in a stable manner to external disturbances. But due to information delay, inherent feedback structure and nonlinear components unstable phenomena can arise which may ultimately manifest as chaotic behavior. Efficient resource allocation and collective oscillations (of say inventory levels) are some examples of emergent behavior shown by supply chains. They have structure at many scales, each component itself represents a simple supply chain. The components compete due to resource limitation but collaborate/cooperate to maximize their gains which is another characteristic feature of a complex system.

The survivability of logistic systems under varying environmental conditions is of paramount importance. Survivability is going to be itself an emergent property of a logistic system and it represents the ability of the system

to function critically even under adverse conditions. We refer to these adverse conditions as stresses. In order to improve the survivability, agents should detect stresses and take appropriate actions so that they can adapt to stress conditions.

Due to lack of analytical tools for predicting emergent behavior of a complex system from its component behavior, simulation is primary tool of designing and optimizing them. In this paper, we would like to show how an agent learns to detect stresses as the first step towards improving survivability. We implemented the supply chain in COUGAAR (Cognitive Agent Architecture developed by DARPA) as a simulation model. Through an extensive design of experiments we subjected the supply chain to various stress conditions and made the agents learn to predict them using Support Vector Machines.

THE SMALL SUPPLY CHAIN (SSC)

We built a multi-agent system for a small supply chain using Cougaar version 8.6.0 (<http://www.cougaar.org>). Cougaar is an open source multi-agent architecture and is appropriate for modelling large-scale distributed supply chain management applications. We call our supply chain system ‘the Small Supply Chain (SSC)’.

Each agent in SSC represents an individual organization such as a retailer and a supplier in the supply chain. Figure 1 represents demand flows in this small supply chain.

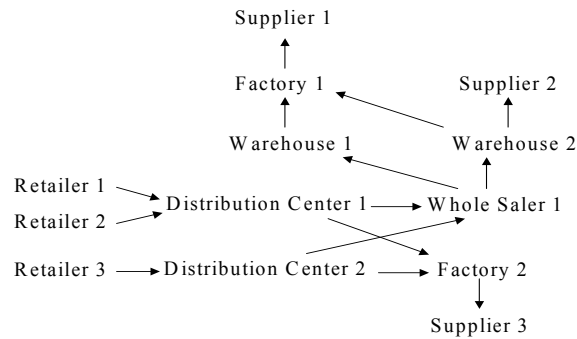


Figure 1. Demand flows in the Small Supply Chain (SSC)

STRESS TYPES AND LEVELS

After some preliminary experiments and observations we used the following stress conditions to show our approach.

Stress 1. Changing OPTEMPO. The SSC works according to a Logistics Plan. The plan for each agent is prespecified. Every activity of each agent has an OPTEMPO value which represents the level of the activity. Changing OPTEMPOs can result in a different plan. OPTEMPO can have one of the three values, ‘low’, ‘medium’ and ‘high’.

Stress 2. Adding and Dropping agents. Dropping agents can represent situations such as communication loss due to physical accidents or cyber attacks. When a retailer agent is dropped, its supporting agent will not receive tasks from the dropped agent and its retailer agents will not receive responses from the

dropped agent. These changes will affect planning significantly. By adding new retailer agents, we can evaluate how sensitive the SSC is to scalability. The addition of new retailer agent increases a load to the other supplier agents.

PREDICTORS

In Cougaar, every agent has its own blackboard. During logistics planning, the intermediate planning results are continuously accumulated on that blackboard. Therefore, by observing the blackboard we can recognize the planning state. Our idea is to detect stresses by observing the blackboard. Each agent should have the ability to detect the stresses coming from outside so that it can make a decision autonomously to handle the stresses.

In this work, for each agent we build a separate supervised learning model. Many types of task classes are instantiated on the blackboard. The collection of the number of tasks of each type represents the state of the agent. A task is a java class of Cougaar which represents a logistic requirement or activity. Tasks are generated successively along the supply chain starting from the tasks of the retailers. The learning model takes the state of the agent as input feature and predicts the corresponding stress type and level. The pattern recognition model - predictor - is built using the Support Vector Machine.

In order to prepare training and test data, the blackboard of each agent is monitored and data is stored into a database during experiments by a monitoring facility which consists of a specialized Plugin and a separate server machine. The Plugin is a java class provided by Cougaar. The pattern recognition model is trained by the data from the database off-line.

SUPPORT VECTOR MACHINES (SVM)

A Support Vector Machine is a pattern recognition method. It has been popular since the mid-90s because of its theoretical clearness and good performance. Many pattern recognition applications have been reported since this theory was developed by Vapnik (Müller, et al., 2001), which also exemplify its superiority over similar such techniques. Moghaddam and Yang (2002) applied SVM to the appearance-based gender classification and showed that it is superior to other classifiers-nearest neighbor, radial basis function classifier. Liang and Lin (2001) showed SVM has better performance than conventional neural network in detection of delayed gastric emptying. For an exhaustive review, we recommend the reader to (Burges, 1998), (Chapelle et al., 1999) and (Müller et al., 2001).

The main idea of SVM is to separate the classes with a surface that maximizes the margin between them. This is an approximate implementation of the Structural Risk Minimization induction principle (Osuna, et al., 1997). To construct a classifier for a given data set, a SVM solves a quadratic programming problem with each variable corresponding to a data point. When the size of the data set is large, it requires special techniques such as decompositions to handle the large number of variables. Basically, the SVM is a linear classifier. Thus in order to handle a dataset which is not separable by a linear function, inner-product kernel functions are used. The role of the inner-product kernel functions is to convert an inner product of low dimensional data

points into a corresponding inner product in high dimensional space without actual mapping. The principle of the mapping is based on Mercer's theorem (Vapnik 1998). By doing so, the SVM overcomes non linear-separable cases.

The selection of kernel functions depends on the problem. We should choose an appropriate function by performing experiments. Other control parameters for the SVM are the extra cost for errors (represented as C), the loss sensitivity constant ($\epsilon_{\text{sensitivity}}$) and the maximum number of iterations. The extra cost for errors, C , is a cost assigned to training errors in non linear-separable cases. A larger C corresponds to assigning a higher penalty to errors (Burges 1998). The loss sensitivity constant ($\epsilon_{\text{sensitivity}}$) represents the allowable error range for the prediction values.

SVMs are basically developed as binary classifiers. Currently a lot of research is being done in the area of multi-class SVM. We use BSVM 2.0 which is the multi-class SVM program suggested by Hsu and Lin (2002).

EXPERIMENT CONDITIONS

We simulated the SSC under various stress conditions. 12 Stress conditions are made through the combination of the following factors; The number of retailer 1(zero, one, two), OPTEMPO of retailer 2 (LOW, HIGH), OPTEMPO of retailer 3 (LOW, HIGH).

219 data sets were used for training and 94 data sets were used for testing the prediction power from a total of 313 data sets. The conditions and number of experiments are shown in the table 1.

Condition	Retailer 1	Retailer 2	Retailer 3	Training	Test	Total
1	Zero	LOW	LOW	25	11	36
2	Zero	HIGH	LOW	19	8	27
3	Zero	LOW	HIGH	18	8	26
4	Zero	HIGH	HIGH	17	7	24
5	One	LOW	LOW	29	13	42
6	One	HIGH	LOW	17	7	24
7	One	LOW	HIGH	18	7	25
8	One	HIGH	HIGH	18	8	26
9	Two	LOW	LOW	21	9	30
10	Two	HIGH	LOW	12	5	17
11	Two	LOW	HIGH	13	6	19
12	Two	HIGH	HIGH	12	5	17
Grand Total	-	-	-	219	94	313

Table 1. The stress condition and number of experiments

TRAINING

Through preliminary studies apart from the experiments tabulated above we found that all the stress conditions do not affect all the agents. Thus, we prepared different classification definitions for the training set depending on the agent (See the table 2).

As the classification definitions are different for different agents, the input features are also different. The tasks used as input features in each agent are shown in table 3.

The option ‘multi-class bound-constrained support vector classification’ in BSVM 2.0 is selected. For others, we use default options of BSVM 2.0 such as the radial basis function with $\gamma = 1/(\text{the number of input features})$.

Agent name	Class definition
Retailer 2, Warehouse 2	Class 1: condition 1,3,5,7,9,11 (Retailer 2 LOW) Class 2: condition 2,4,6,8,10,12 (Retailer 2 HIGH)
Retailer 1	Class 1: condition 1,2,3,4 (zero Retailer 1) Class 2: condition 5,6,7,8 (one Retailer 1) Class 3: condition 9,10,11,12 (two Retailer 1)
Factory 2, Supplier 2	Class 1: condition 1,3 (Retailer 2 LOW at zero Retailer 1) Class 2: condition 2,4 (Retailer 2 HIGH at zero Retailer 1) Class 3: condition 5,6,7,8,9,10,11,12 (All other cases)
Warehouse 1, Factory 1	12 Classes, Regard each condition as one class
Retailer 3, Distribution Center 2	Class 1: condition 1,2,5,6,9,10 (LOW Retailer 3) Class 2: condition 3,4,7,8,11,12 (HIGH Retailer 3)
Distribution Center 1, Supplier 1	Class 1: 1,2 (LOW Retailer 3 at zero Retailer 1) Class 2: 3,4 (HIGH Retailer 3 at zero Retailer 1) Class 3: 5,6 (LOW Retailer 3 at one Retailer 1) Class 4: 7,8 (HIGH Retailer 3 at one Retailer 1) Class 5: 9,10 (LOW Retailer 3 at two Retailer 1) Class 6: 11,12 (HIGH Retailer 3 at two Retailer 1)
Wholesaler 1	Class 1: condition 9,10,11,12 (Two Retailer 1) Class 2: condition 1,2,3,4,5,6,7,8 (other conditions)

Table 2. The class definition by agents

Agent	Features	Agent	Features
Retailer 2	PS, PW	Warehouse 1	PS, PW, OS
Distribution Center 1	W, OPS, OS	Factory 1	TP, W, OPS, OS
Retailer 3	PS, PW	Supplier 1	TR, TP, OTP
Distribution Center 2	PS, PW	Retailer 1	PS, PW
Factory 2	TP	Warehouse 2	TP, W
Supplier 2	OTP	Wholesaler 1	S

* PS = ProjectSupply, PW = ProjectWithdraw, W =Withdraw, TP = Transport,
TR = Transit, S = Supply, OPS = ProjectSupply coming from outside,
OS = Supply coming from outside, OTP = Transport coming from outside

Table 3. The input features by agents

Agent	Success rate	Agent	Success rate
Retailer 2	100%	Warehouse 1	100%
Retailer 1	100%	Distribution Center 1	100%
Retailer 3	100%	Factory 1	22.34%
Distribution Center 2	100%	Supplier 1	40.43%
Factory 2	100%	Warehouse 2	84.04%
Supplier 2	100%	Wholesaler 1	86.17%

Table 4. The success rate to classify the stress condition at each agent

RESULTS

The table 4 contains the test results. Overall performance is good. In addition, we can see the agent near the retailers in the supply chain can detect

stresses well. The Warehouse 1 agent can detect exactly all the stress types even though it is far from retailers (see Fig 1.).

CONCLUSIONS

We have shown an effective application of pattern recognition model for detecting stresses by observing the internal state of each agent. Each agent has a SVM since the influence of the same stress on different agents can be different. Some agents near the retailers can detect stresses very well. However, it is hard to detect the influence of the stress on agents which are far from retailers. Overall performance of predictor of each agent is good. Constructing the capability for stress detection is the first step towards improving the survivability of a multi-agent system. This result is important because we can pursue further research on how we can dampen the effect of stresses based on the result of this study. Based on current detected state each agent can change their behaviors - ordering or planning - to adapt to stress conditions without serious performance degradation of overall supply chain. In addition, our approach is generally useful because it is very hard to model a complex system analytically.

ACKNOWLEDGEMENTS

Support for this research was provided by DARPA (Grant#: MDA 972-01-1-0563) under the UltraLog program.

REFERENCES

- Baranger, Michel, "Chaos, Complexity, and Entropy – A physics talk for non-physicists," MIT-CTP-3112, <http://necsi.org/projects/baranger/cce.pdf>.
- Burges, C. J. C., 1998, "A Tutorial on Support Vector Machines for Pattern Recognition," *Knowledge Discovery and Data Mining*, Vol. 2, No. 2, pp. 121-167.
- Chapelle, O., Haffner, P. and Vapnik, V. N., 1999, "Support Vector Machines for Histogram-Based Image Classification," *IEEE Transactions on Neural Networks*, Vol. 10, No. 5, pp. 1055-1064.
- Choi, T., Dooley, K. and Rungtusanatham, M., 2000, "Supply Networks and Complex Adaptive Systems: control versus emergence," *Journal of Operations Management*, Vol. 19, pp 351-366.
- Dooley, K., 2002, "Simulation Research Methods," Companion to Organizations, Joel Baum (ed.), London: Blackwell, pp. 829-848.
- Hsu, Chih-Wei and Lin, Chih-Jen, 2002, "A Comparison of Methods for Multiclass Support Vector Machines," *IEEE Transactions on Neural Networks*, Vol. 13, No. 2, pp. 415-425.
- Liang, H. and Lin, Z., 2001, "Detection of Delayed Gastric Emptying from Electrogastrograms with Support Vector Machine," *IEEE Transactions on Biomedical Engineering*, Vol. 13, No. 2, pp. 415-425.
- Moghaddam, B. and Yang, M., 2002, "Learning Gender with Support Faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 5, pp. 707-711.
- Müller, K., Mika, S., Rätsch, G., Tsuda, K. and Schölkopf B., 2001, "A Introduction to Kernel-Based Learning Algorithms," *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pp. 181-202.
- Osuna, E. E., Freund R. and Girosi, F., 1997, Support Vector Machines: Training and Applications, Technical Report AIM-1602, MIT A.I. Lab.
- Vapnik, V. N., 1998, *Statistical Learning Theory*, John Wiley & Sons, Inc, New York.