8th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
S u c e a v a, R o m a n i a, M a y 25 – 27, 2 0 0 6

LOAD BALANCING IN MOBILE INTELLIGENT AGENTS FRAMEWORK USING DATA MINING CLASSIFICATION TECHNIQUES

Florin LEON, Butincu CRISTIAN, Mihai Horia ZAHARIA
Technical University "Gh.Asachi" Iasi
Bd-ul Mangeron nr. 53A, RO-700050 Iasi
mike@cs.tuiasi.ro

Abstract. *In This paper a new approach in solving load balancing problem in distributed inteligents framewoks were presented. This will use time series prediction. Instead of usual approach in artificial intelligence that use neural networks or neuro-fuzzy approach we prefer to analyze the temporal decision trees comportment. As result we prove that very good results can be obtained using data mining algorithms for classification if the data are interpreted in a temporal way.*
Keywords: *load balancing, intelligent agents, data mining, distributed systems*

Introduction

The load balancing problem appears from the first parallel computing machines. It still remains young. The contradictory sets of constraints that are specific to the problem drive all the research to this conclusion [11]. It remains also a problem for the latest approach in distributed computing that consists in using intelligent agent's frameworks.
The framework of agents has brought together many disciplines in an effort to build distributed, intelligent, and robust applications. They have given us a new way to look at distributed systems and provided a path to more robust intelligent applications. Unfortunately, many of the methods focused on single agent architecture or have not gone to the necessary level of detail to adequately support complex system development [3].
At this hour we have two types of approach in load balancing. One refers to the classical algorithms use in load balancing. A typical example is DASUD (Diffusion Algorithm Searching Unbalanced Domains) which is flexible in terms of allowing one to control the balancing qualities, effective for preserving communication locality and easily scaled in parallel computers with a direct communication network, that was proposed in [1,2]. At the theoretical level classical approach seems to be suitable. Unfortunately good results are obtained in homogenous networks because of the model simplicity. At the heterogeneous computing level this algorithm begin have problem onto solving real time load balancing problems due to the complex calculations required.
When the artificial intelligence begins to develop seems that they can solve adequately a wide area of problems. But they have increased complexity than classical approaches in many cases. Until the distributed computing begin to be reliable the stay into shade as applicability in some domains. At this hour they begin to gain a new life. In this paper we analyze temporal decision trees comportment in solving load balancing at the framework level. In order to do that dedicated agents will be used

Load prediction as time series prediction

If one is to try to predict a system load at a certain time, the general load function can be seen as a time series. A classical way of dealing with time series prediction is by using neural networks [5]. The negative side of using neural networks is the fact that they are sub-symbolic "black boxes", and their states cannot be easily inspected.
The use of load prediction can be found if one considers a mobile agent that aims at executing different tasks on different machines. It can sense its environment as the load of the system and execute only if the load is lower than a specified threshold. However, it would be better to use a form of previous experience in order to predict the load on a certain machine before

dispatching to it. Given a moment in time, the agent will try to determine the load of the system at the next moment.

In the following, we attempt to solve the time series prediction problem using different classification methods from the data mining field. The decision tree approach was presented in previous studies using the C4.5 algorithm [4,5]. Since the load in a uni-dimensional function, it is necessary to preprocess the data in order to use the standards C4.5 algorithm. This process is known as "flattening" the data, thus transforming the simple (x, $f(x)$) pairs into temporal decision data. The method is very simple: given the function values at different consecutive moments of time t_1, t_2, the temporal transaction will be of the form: $\left(t_1, f(t_1), t_2, f(t_2)\right)$, where $f(t_2)$ should be interpreted as the class to be discovered through applying the data mining algorithm.

In this case, we have a time window of size 2 for prediction. The prediction at a certain time depends only on the previous value of the function. However, the latter in its turn depends on the previous and thus the whole time series can be foretold.

The results must be also interpreted in a different way from the classical approach. In the temporal decision tree or temporal classification context in general, the predicted class in the predicted function value at the specified time.

Mobile agents' framework

The mobile agents' framework provides the end-user with possibility to easily build a virtual cluster that fits its needs and also to find and use existing services. This virtual cluster guarantees to contain only the nodes that are capable of overall performance increase. The framework was entirely developed using the Java programming language.

The main network, to which each user has access to, is composed by a set of POPs (Point of Presence). Each POP is actually a server that provides the user with an access point to the network. So, the only thing the user needs to know at a given time is the name of the nearest POP. A POP is the entry point of the user in system. Every POP in the network keeps a list of its nearest links (nodes). Once the agent arrives at the POP, he may query that node during which it receives the POP's list of links. The framework design permits the possibility to give different list of nodes to different agents based on the category from which the agent is being part of. At any time the agent can start a trip to any of the nearest nodes.

In order to provide the node search and reserve paradigm, several ideas were taken into consideration. Some of the questions were how and when the trip agent should contact and communicate results back to the user. One idea was that the trip agent to visit all the hosts on its itinerary and at the end to return with these results back to the origin. The problem here is that the agent can be trapped inside a host and all its information up to that point would be lost. The framework supports the concept of "one way" messages, that is, the trip agent doesn't need to know and moreover, it doesn't care if the messages arrive at its origin or not. Once the trip agent starts its trip and once it arrives on a node, it checks that node to see if it's suitable to be reserved and if so, it sends a "one way" message with the information about this node back to its origin. If this message arrives or not to the origin isn't an issue for this trip agent. He keeps walking through the nodes repeating these steps until he reaches its final destination. During its trip, the agent also stores the nodes information inside in its data structures. This approach acts like a backup solution. At its final destination, he already has all information inside its structures. At this point the agent can send a regular message with all this information at once back to its origin.

To illustrate the benefits of this approach let's assume the following scenario: the agent starts its trip and at a given time it ends-up with a huge list of destinations. The user definitely should not wait for that agent to check all nodes in the list. By using "one way" messages part or maybe all messages will arrive at the origin providing the user with the possibility of interacting with those nodes. If at some point the subsystem on which the trip agent happens to be, fails and crashes-down there is no problem;

the user wouldn't have to face with the lack of information anymore, because at that point he will still have some nodes (if not all, only part of them) to work with. Moreover, the user wouldn't have to wait for the agent to inspect all nodes. He can start his interaction with the system as soon as the nodes are found. The framework design permits implementation of a wide range of paradigms; for example a population of trip agents that can use agent clones can be implemented.

The framework provides support for developing any kind of agent like: load balancing agents, security agents, communication agents, mobile or stationary agents, intelligent agents or state agents and so on.

As for the communication protocol, this framework uses RMI (Remote Method Invocation).

Firewall issues were also taken into consideration: to overcome firewalls the servers that make up the framework can be configured to run on any port of choice so http port 80 can be used.

Each server in the framework that runs in a single JVM (Java Virtual Machine) is architected in such a way that can host multiple agencies. This approach can lead to easy classification of agents into agencies and of agencies. Such classifications are limitless and can range from commercial agencies to free agencies to research agencies to topic agencies.

Load balancing, fault tolerance and security are separate modules that ca be plugged in on top of this framework. As for load balancing we rely on load prediction with mobile intelligent agents using data mining classification techniques. This approach is detailed in the next sections of this article.

Case studies

In order to apply the classification methods for time series prediction, we considered the function displayed in figure 1, which resulted from running an agent of the mobile agent framework on a system. From the implementation point of view, we used the variants described in the data mining book by Witten and Frank [10].

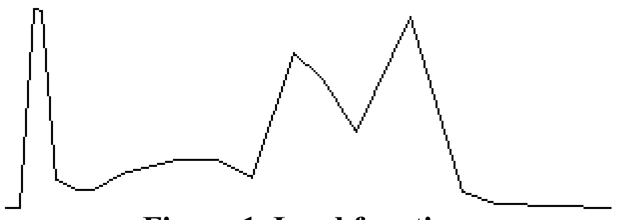

Figure 1. Load function.

One classification method is that using non-nested generalized exemplars [8,9]. It is an instance-based learning that reduces the learning effort by simply storing the examples presented to the learning agent, and classifying the new instances on the basis of closeness to their "neighbors", i.e. previously encountered instances with similar attribute value.

When the loads are discretized into 5 classes (very-low, low, medium, high, very-high), the hyper-rectangles discovered by the algorithm in the form of temporal rules, for a time window of size 2, are presented in Appendix A.

Prediction accuracy, taking into account the discretization of continuous values, is 100%.

Another classification method is the regression by discretization [10]. The regression by discretization approach allows the use of classification algorithm in a regression task. It works as a pre-processing step in which the numeric target value is discretized into a set of intervals. A regression scheme that employs any classifier on a copy of the data that has the class attribute (equal-width) discretized. The predicted value is the expected value of the mean class value for each discretized interval (based on the predicted probabilities for each interval).

The results of regression by discretization using a C4.5 classifier and 10 classes [5] are presented below in the form of a decision tree in Appendix B. The results are also graphically displayed in figure 2. The C4.5 algorithm generates a classification-decision tree for the given dataset by recursive partitioning of data. The decision is grown using depth-first strategy. The algorithm considers all the possible tests that can split the data set and selects a test that gives the best information gain. For each discrete attribute, one test with outcomes as many as the number of distinct values of the attribute is considered. For each continuous attribute, binary tests involving every distinct values of the attribute are considered. In order to gather the entropy gain of all these binary tests efficiently, the training data set belonging to the node in consideration is sorted for the values of the continuous attribute and the entropy gains of the binary cut based on each distinct values are calculated in one scan of the sorted data. This process is repeated for each continuous attribute [7].

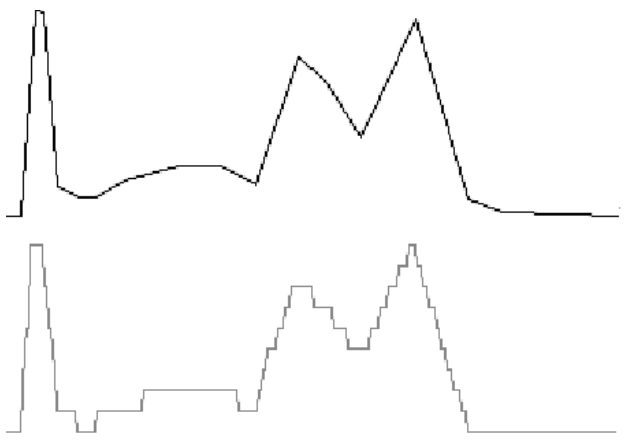

Figure 2. Prediction using regression by discretization with C4.5.

The mean absolute error is this case is 2.7647, and the root mean squared error is 3.5691. The shape of the predicted function follows the original function closely; in this case the correlation coefficient is 0.9919.
In the following, we employed a larger time window, of size 4, i.e. predicting the current value of the function by taking into account not only the previous time moment and function value (load), but 3 previous pairs of moments/loads. The temporal transactions (discretized into 5 classes, as before) are in the form of: (0, *VeryLow*, 1, *VeryLow*, 2, *VeryLow*, 3, *VeryLow*). The non-nested generalized exemplars algorithm provides the following results (hyper-rectangles or rules) as can be seened in Appendix C.

In this case, one can notice that the rules are more complex, but the prediction capability is 100% on the training set, just like before.
The random tree classifier was proposed by Frank and Kirkby [10]. It builds a tree that considers *k* random features at each node, and performs no pruning; therefore its error rate on the training set alone is rather small. In our case, the error rate on the training set in 0%. A more complex decision tree given by the random tree algorithm is presented in Appendix D.
Again, the error on the training set (i.e. the function to be predicted) is 100%, although the size of the tree is rather large, due to both the classification method (random attribute tests) and the size of the time window considered.

Conclusions

In this paper a new approach in solving load balancing problems is presented. The used algorithm it is known as time consuming one but the use of distributed computing approach overcomes this problem. We show that time series prediction can be done in a different way than using the neural network approach. More, very good results can be obtained using data mining algorithms for classification, provided that the data are interpreted in a temporal way. The great advantage of the method is that well established classification algorithms (which can be sometimes very complex) can be used without any modifications, just with simple pre-processing and post-processing of data.

References

[1]. 1. Cortés A., Ripoll A., Senar M. A. and Luque E. (1997), Dynamic Load Balancing Strategy for Scalable Parallel Systems", PARCO'97, 1997.
[2]. Cortés A., Ripoll A., Senar M. A. , Cedó F. and E. Luque (1998), "On the convergence of SID and DASUD load-balancing algorithms", Technical Report, UAB.
[3]. Iglesias C., Garijo M., and Gonza'lez J., "A survey of agent-oriented methodologies", in Intelligent Agents V. Agents Theories, Architectures, and Languages, Lecture Notes, in

Computer Science, vol. 1555, eds. J. P. Mu¨ller, M. P. Singh, and A. S. Rao (Springer- Verlag, 1998).

[4]. Karimi, K. Hamilton, H. J. (2001). *Temporal Rules and Temporal Decision Trees: A C4.5 Approach*, Technical Report CS-2001-02, University of Regina, Saskatchewan, Canada.

[5]. Quinlan, R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA.

[6]. Leon, F. , Gâlea, D., Zaharia, M. H. (2002). *Load Balancing in Distributed Systems using Cognitive Behavioral Models*, Bulletin of Technical University of Iasi, tome XLVIII (LII), fasc. 1-4.

[7]. Leon, F., Zaharia, M.H., Gâlea, D. (2004). *Performance Analysis of Categorization Algorithms*, in Proceedings of the 8th International Symposium on Automatic Control and Computer Science, Iasi.

[8]. Leon, F. (2005). *New Classes of Intelligent Agents with Cognitive Capabilities*, PhD Thesis, “Gh. Asachi” Technical University of Iasi, Romania.

[9]. Martin, B. (1995). *Instance-Based learning: Nearest Neighbor With Generalization*, Master Thesis, University of Waikato, Hamilton, New Zealand.

[10]. Witten, I. H., Frank, E. (2000). *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco.

[11]. Zaharia M. H., Leon F., Gâlea D., *Parallel Genetic Algorithms for Cluster Load Balancing*, Proceedings of the European Conference on Intelligent Systems and Technologies, Iasi, July 2004, ISBN 973-7994-78-7

Appendix A

```
class VeryLow IF : 225.0<=t1<=300.0 OR f1 in {VeryLow,Low} OR 226.0<=t2<=301.0
class VeryLow IF : 0.0<=t1<=6.0 OR f1 in {VeryLow} OR 1.0<=t2<=7.0
class VeryLow IF : 25.0<=t1<=62.0 OR f1 in {VeryLow,Low} OR 26.0<=t2<=63.0
class VeryLow IF : 116.0<=t1<=122.0 OR f1 in {VeryLow,Low} OR 117.0<=t2<=123.0
class Low IF : 219.0<=t1<=224.0 OR f1 in {Low,Medium} OR 220.0<=t2<=225.0
class Low IF : 7.0<=t1<=8.0 OR f1 in {VeryLow,Low} OR 8.0<=t2<=9.0
class Low IF : 23.0<=t1<=24.0 OR f1 in {Low,Medium} OR 24.0<=t2<=25.0
class Low IF : 63.0<=t1<=115.0 OR f1 in {VeryLow,Low} OR 64.0<=t2<=116.0
class Low IF : 123.0<=t1<=129.0 OR f1 in {VeryLow,Low} OR 124.0<=t2<=130.0
class Medium IF : 213.0<=t1<=218.0 OR f1 in {Medium,High} OR 214.0<=t2<=219.0
class Medium IF : 21.0<=t1<=22.0 OR f1 in {Medium,High} OR 22.0<=t2<=23.0
class Medium IF : 130.0<=t1<=136.0 OR f1 in {Low,Medium} OR 131.0<=t2<=137.0
class Medium IF : 161.0<=t1<=183.0 OR f1 in {Medium,High} OR 162.0<=t2<=184.0
class Medium IF : t1=9.0 OR f1 in {Low} OR t2=10.0
class High IF : 207.0<=t1<=212.0 OR f1 in {High,VeryHigh} OR 208.0<=t2<=213.0
class High IF : 10.0<=t1<=11.0 OR f1 in {Medium,High} OR 11.0<=t2<=12.0
class High IF : 137.0<=t1<=160.0 OR f1 in {Medium,High} OR 138.0<=t2<=161.0
class High IF : 184.0<=t1<=192.0 OR f1 in {Medium,High} OR 185.0<=t2<=193.0
class High IF : t1=20.0 OR f1 in {VeryHigh} OR t2=21.0
class VeryHigh IF : 193.0<=t1<=206.0 OR f1 in {High,VeryHigh} OR 194.0<=t2<=207.0
class VeryHigh IF : 12.0<=t1<=19.0 OR f1 in {High,VeryHigh} OR 13.0<=t2<=20.0
```

Appendix B

```
f1 <= 34
|   f1 <= 10: '(-inf-10.9]'
|   f1 > 10
|   |   f1 <= 20: '(10.9-20.8]'
|   |   f1 > 20
|   |   |   f1 <= 27: '(20.8-30.7]'
|   |   |   f1 > 27
|   |   |   |   t1 <= 174: '(30.7-40.6]'
|   |   |   |   t1 > 174: '(20.8-30.7]'
f1 > 34
|   f1 <= 60
|   |   f1 <= 48
|   |   |   t1 <= 217: '(40.6-50.5]'
|   |   |   t1 > 217: '(30.7-40.6]'
|   |   f1 > 48
|   |   |   t1 <= 214: '(50.5-60.4]'
|   |   |   t1 > 214: '(40.6-50.5]'
|   f1 > 60
|   |   f1 <= 77
|   |   |   f1 <= 68
|   |   |   |   t1 <= 79: '(50.5-60.4]'
|   |   |   |   t1 > 79: '(60.4-70.3]'
|   |   |   f1 > 68
|   |   |   |   t1 <= 151: '(70.3-80.2]'
|   |   |   |   t1 > 151
|   |   |   |   |   t1 <= 171: '(60.4-70.3]'
|   |   |   |   |   t1 > 171
|   |   |   |   |   |   t1 <= 208: '(70.3-80.2]'
|   |   |   |   |   |   t1 > 208: '(60.4-70.3]'
|   |   f1 > 77
|   |   |   t1 <= 18: '(90.1-inf)'
|   |   |   t1 > 18
|   |   |   |   t1 <= 193: '(70.3-80.2]'
|   |   |   |   t1 > 193
|   |   |   |   |   t1 <= 205
|   |   |   |   |   |   f1 <= 88: '(80.2-90.1]'
|   |   |   |   |   |   f1 > 88
|   |   |   |   |   |   |   t1 <= 202: '(90.1-inf)'
|   |   |   |   |   |   |   t1 > 202: '(80.2-90.1]'
|   |   |   |   |   t1 > 205: '(70.3-80.2]'
```

Appendix C

```
        class VeryLow IF : 223.0<=t1<=299.0 OR f1 in {VeryLow,Low} OR 224.0<=t2<=300.0 OR f2 in
{VeryLow,Low} OR 225.0<=t3<=301.0 OR f3 in {VeryLow,Low} OR 226.0<=t4<=302.0
        class VeryLow IF : 23.0<=t1<=60.0 OR f1 in {VeryLow,Low,Medium} OR 24.0<=t2<=61.0 OR f2 in
{VeryLow,Low} OR 25.0<=t3<=62.0 OR f3 in {VeryLow,Low} OR 26.0<=t4<=63.0
        class VeryLow IF : 114.0<=t1<=120.0 OR f1 in {VeryLow,Low} OR 115.0<=t2<=121.0 OR f2 in
{VeryLow,Low} OR 116.0<=t3<=122.0 OR f3 in {VeryLow,Low} OR 117.0<=t4<=123.0
        class VeryLow IF : 0.0<=t1<=4.0 OR f1 in {VeryLow} OR 1.0<=t2<=5.0 OR f2 in {VeryLow} OR
2.0<=t3<=6.0 OR f3 in {VeryLow} OR 3.0<=t4<=7.0
        class Low IF : 217.0<=t1<=222.0 OR f1 in {Low,Medium} OR 218.0<=t2<=223.0 OR f2 in {Low,Medium}
OR 219.0<=t3<=224.0 OR f3 in {Low,Medium} OR 220.0<=t4<=225.0
        class Low IF : 61.0<=t1<=113.0 OR f1 in {VeryLow,Low} OR 62.0<=t2<=114.0 OR f2 in {VeryLow,Low}
OR 63.0<=t3<=115.0 OR f3 in {VeryLow,Low} OR 64.0<=t4<=116.0
        class Low IF : 121.0<=t1<=127.0 OR f1 in {VeryLow,Low} OR 122.0<=t2<=128.0 OR f2 in
{VeryLow,Low} OR 123.0<=t3<=129.0 OR f3 in {VeryLow,Low} OR 124.0<=t4<=130.0
        class Low IF : 5.0<=t1<=22.0 OR f1 in {VeryLow,Medium,High} OR 6.0<=t2<=23.0 OR f2 in
{VeryLow,Medium} OR 7.0<=t3<=24.0 OR f3 in {VeryLow,Low,Medium} OR 8.0<=t4<=25.0
        class Medium IF : 211.0<=t1<=216.0 OR f1 in {Medium,High} OR 212.0<=t2<=217.0 OR f2 in
{Medium,High} OR 213.0<=t3<=218.0 OR f3 in {Medium,High} OR 214.0<=t4<=219.0
        class Medium IF : 128.0<=t1<=134.0 OR f1 in {Low,Medium} OR 129.0<=t2<=135.0 OR f2 in
{Low,Medium} OR 130.0<=t3<=136.0 OR f3 in {Low,Medium} OR 131.0<=t4<=137.0
        class Medium IF : 159.0<=t1<=181.0 OR f1 in {Medium,High} OR 160.0<=t2<=182.0 OR f2 in
{Medium,High} OR 161.0<=t3<=183.0 OR f3 in {Medium,High} OR 162.0<=t4<=184.0
        class Medium IF : t1=7.0 OR f1 in {VeryLow} OR t2=8.0 OR f2 in {Low} OR t3=9.0 OR f3 in {Low}
OR t4=10.0
        class Medium IF : 19.0<=t1<=20.0 OR f1 in {VeryHigh} OR 20.0<=t2<=21.0 OR f2 in {High,VeryHigh}
OR 21.0<=t3<=22.0 OR f3 in {Medium,High} OR 22.0<=t4<=23.0
        class High IF : 205.0<=t1<=210.0 OR f1 in {High,VeryHigh} OR 206.0<=t2<=211.0 OR f2 in
{High,VeryHigh} OR 207.0<=t3<=212.0 OR f3 in {High,VeryHigh} OR 208.0<=t4<=213.0
        class High IF : 135.0<=t1<=158.0 OR f1 in {Medium,High} OR 136.0<=t2<=159.0 OR f2 in
{Medium,High} OR 137.0<=t3<=160.0 OR f3 in {Medium,High} OR 138.0<=t4<=161.0
        class High IF : 182.0<=t1<=190.0 OR f1 in {Medium,High} OR 183.0<=t2<=191.0 OR f2 in
{Medium,High} OR 184.0<=t3<=192.0 OR f3 in {Medium,High} OR 185.0<=t4<=193.0
        class High IF : 8.0<=t1<=9.0 OR f1 in {Low} OR 9.0<=t2<=10.0 OR f2 in {Low,Medium} OR
10.0<=t3<=11.0 OR f3 in {Medium,High} OR 11.0<=t4<=12.0
        class High IF : t1=18.0 OR f1 in {VeryHigh} OR t2=19.0 OR f2 in {VeryHigh} OR t3=20.0 OR f3 in
{VeryHigh} OR t4=21.0
        class VeryHigh IF : 191.0<=t1<=204.0 OR f1 in {High,VeryHigh} OR 192.0<=t2<=205.0 OR f2 in
{High,VeryHigh} OR 193.0<=t3<=206.0 OR f3 in {High,VeryHigh} OR 194.0<=t4<=207.0
        class VeryHigh IF : 10.0<=t1<=17.0 OR f1 in {Medium,High,VeryHigh} OR 11.0<=t2<=18.0 OR f2 in
{High,VeryHigh} OR 12.0<=t3<=19.0 OR f3 in {High,VeryHigh} OR 13.0<=t4<=20.0
```

Appendix D

```
f1 = VeryLow
|   t4 < 177.5
|   |   f3 = VeryLow
|   |   |   t1 < 120.5
|   |   |   |   t2 < 61.5
|   |   |   |   |   t4 < 18.5
|   |   |   |   |   |   t3 < 6.5 : VeryLow
|   |   |   |   |   |   t3 >= 6.5 : Low
|   |   |   |   |   t4 >= 18.5 : VeryLow
|   |   |   |   t2 >= 61.5
|   |   |   |   |   t4 < 92 : Low
|   |   |   |   |   t4 >= 92 : VeryLow
|   |   |   t1 >= 120.5 : Low
|   |   f3 = Low
|   |   |   t2 < 35.5
|   |   |   |   t4 < 9.5 : Low
|   |   |   |   t4 >= 9.5 : Medium
|   |   |   t2 >= 35.5 : Low
|   |   f3 = Medium : VeryLow
|   |   f3 = High : VeryLow
|   |   f3 = VeryHigh : VeryLow
|   t4 >= 177.5 : VeryLow
f1 = Low
|   f3 = VeryLow : VeryLow
|   f3 = Low
|   |   t3 < 115.5 : Low
|   |   t3 >= 115.5
|   |   |   t1 < 119 : VeryLow
|   |   |   t1 >= 119
|   |   |   |   t1 < 222.5
|   |   |   |   |   t2 < 128.5 : Low
|   |   |   |   |   t2 >= 128.5
|   |   |   |   |   |   t2 < 175 : Medium
|   |   |   |   |   |   t2 >= 175 : Low
|   |   |   |   t1 >= 222.5 : VeryLow
|   f3 = Medium
|   |   t2 < 69.5 : High
|   |   t2 >= 69.5 : Medium
|   f3 = High : High
|   f3 = VeryHigh : VeryLow
f1 = Medium
|   t4 < 80
|   |   f2 = VeryLow : VeryLow
|   |   f2 = Low : VeryLow
|   |   f2 = Medium : Low
|   |   f2 = High : VeryHigh
|   |   f2 = VeryHigh : VeryLow
|   t4 >= 80
|   |   t3 < 218.5
|   |   |   t2 < 182.5
|   |   |   |   f3 = VeryLow : VeryLow
|   |   |   |   f3 = Low : VeryLow
|   |   |   |   f3 = Medium
|   |   |   |   |   t1 < 148.5
|   |   |   |   |   |   t1 < 134.5 : Medium
|   |   |   |   |   |   t1 >= 134.5 : High
|   |   |   |   |   t1 >= 148.5 : Medium
|   |   |   |   f3 = High : High
|   |   |   |   f3 = VeryHigh : VeryLow
|   |   |   t2 >= 182.5
|   |   |   |   t2 < 200 : High
|   |   |   |   t2 >= 200 : Medium
|   |   t3 >= 218.5 : Low
f1 = High
|   t1 < 79.5
|   |   t3 < 18.5 : VeryHigh
|   |   t3 >= 18.5 : Low
|   t1 >= 79.5
|   |   t4 < 161.5 : High
|   |   t4 >= 161.5
|   |   |   t3 < 175 : Medium
|   |   |   t3 >= 175
|   |   |   |   t3 < 212.5
|   |   |   |   |   f2 = VeryLow : VeryLow
|   |   |   |   |   f2 = Low : VeryLow
|   |   |   |   |   f2 = Medium : VeryLow
|   |   |   |   |   f2 = High
|   |   |   |   |   |   t4 < 193.5 : High
|   |   |   |   |   |   t4 >= 193.5
|   |   |   |   |   |   |   t2 < 201 : VeryHigh
|   |   |   |   |   |   |   t2 >= 201 : High
|   |   |   |   |   f2 = VeryHigh : VeryHigh
|   |   |   |   t3 >= 212.5 : Medium
f1 = VeryHigh
|   f3 = VeryLow : VeryLow
|   f3 = Low : VeryLow
|   f3 = Medium : Medium
|   f3 = High
|   |   t3 < 114.5 : Medium
|   |   t3 >= 114.5 : High
|   f3 = VeryHigh
|   |   t2 < 205.5
|   |   |   t4 < 109
|   |   |   |   t1 < 17.5 : VeryHigh
|   |   |   |   t1 >= 17.5 : High
|   |   |   t4 >= 109 : VeryHigh
|   |   t2 >= 205.5 : High
```