Input Data Management in Real-Time AI Systems

Richard Washington and Barbara Hayes-Roth
Knowledge Systems Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

A real-time AI system in the real world needs to monitor an immense volume of data. To do this, the system must filter out much of the incoming data. However, it must remain responsive to important or unexpected events in the data. This paper describes some simple approaches to data management, shows how they can fail to be both adequately selective and responsive, and presents an approach that improves on the simple approaches by making use of information about the system's resources and ongoing tasks. The new approach has been applied in a system for monitoring patients in a surgical intensive-care unit.

1 Introduction

When an AI system meets the real world, it is confronted by an overwhelming amount of data. To maintain real-time performance, the system must reduce the flow of data to a manageable amount. Some simple approaches have been used to filter the incoming data, but these have severe shortcomings. To operate in real time while remaining responsive to important external events, a system needs a more intelligent approach to data management.

The real world supplies a continuous stream of data for a system to monitor. The system has sensors that sample the data stream at a high enough rate to catch all of the data that the system needs. However, no system doing complex processing can keep up with the nearly-continuous data stream from many sensors. Some sort of data reduction is necessary.

The problem is not limited to reducing a continuous data stream, however. Some sensors may notify a system of particular events in the world rather than transmitting a continuous stream of data. For instance, there may be a sensor that notices a person walking into a room. Unless it had knowledge about when such an event would happen, a system could not accurately predict the event. If it were receiving data from this sensor, the system would need to be able to react quickly to unexpected or unpredictable events.

Also, without a perfect model of the world, the value of any data point is unpredictable. The better the system's model of the world, the better a prediction it could make, but it must be prepared for some unexpected data values. These values should not be ignored, since they might indicate an important event that the system should handle.

A real-time AI system must be able to reduce the data stream to a manageable amount while remaining responsive to unexpected data arrival and values. The object of this paper is to show methods for intelligent data management that address these issues. First, some simple data reduction approaches will be discussed. These methods have been used in existing real-time systems, but they are inadequate to handle the kind of variability actually present in the real world. Second, an approach will be introduced that makes use of more intelligent data management techniques, making use of knowledge about the system's resources and current tasks. Finally, this approach will be shown as it works in Guardian, a system for monitoring patients in a surgical intensive-care unit.

2 Limitations of Current Approaches

The *fixed-sampling* approach to data reduction involves fixed-time sampling of the sensor data. This method uses an explicit, preset sample rate [Andersson, 1988, Fagan *et al.*, 1984] where the system checks the input once every time interval. Two assumptions are implicit in this approach. First, the system must be able to complete its processing of one set of input before the next set arrives. However, the system may remain needlessly idle when it is performing little reasoning, since there may be interesting data between the sampled data. Conversely, the system may be too slow when the data require a large amount of reasoning. Second, the samples may not accurately reflect the actual data. For example, see Figure 1, where the sampled data show little of what is actually happening.

One variant of the fixed sampling approach is the *averaged-sampling* approach, which uses the average value of a parameter over the sampling interval. This approach has mainly been used for parameters where instantaneous values either don't exist or are highly inaccurate, such as *heart rate* in [Fagan *et al.*, 1984]. This still suffers from the problems of a fixed time interval, but it begins to address the problem of errors due to data fluctuation. However, much of the detail of the data is still lost, as can be seen in Figure 2.

Another sampling method, *polling,* trades the difficulties of a fixed time interval for less reliable data. Under this approach, the system samples the data, performs whatever reasoning necessary, and then repeats the procedure [Kaemmerer and Allard, 1987, Nitao and Parodi, 1985]. Here the system is guaranteed to be neither idle nor overwhelmed by data, since it reads data whenever it has processed the previous data. But all the problems with fixed sampling in losing the detail of the data are compounded. Whereas fixed sampling guarantees that any interesting data events lasting longer than the sampling interval will be noticed, polling's interval is dependent on the processing. This variable interval makes the behavior of the system less well-defined. Figure 3 shows an example of such a system's behavior, where in this case the amount of reasoning depends on the magnitude of the data point.

A final approach, *fixed thresholding,* uses fixed thresholds to signal out-of-range data, generating interrupts for the system to handle [Ali and Scharnhorst, 1985, Anderson *et al*., 1984, Chen, 1985]. This requires the data to be well-behaved with respect to the thresholds chosen. Variations just within the allowable range will escape notice, even when the system is idle. Slightly wider variations that cross the threshold may cause multiple interrupts. In fact, one of the worst situations for a fixed threshold is when the value hovers near the threshold. Figure 4 shows how fixed thresholding can cause both periods of inactivity and periods of overabundant data.

3 Intelligent Data Management

The characteristics of real-world data and the shortcomings of the simple methods for data management suggest a set of criteria for more effective data management:

* The system should be responsive to changing resource requirements of the reasoning system. The amount of data sampled should depend on how much time the system needs to process those data.
♦ The system should be responsive to important events in the data. The fact that the system is busy should not prevent it from noticing crucial data.
♦ The system should be able to focus its attention dynamically. If some parameters are particularly relevant to the current reasoning of the system, they

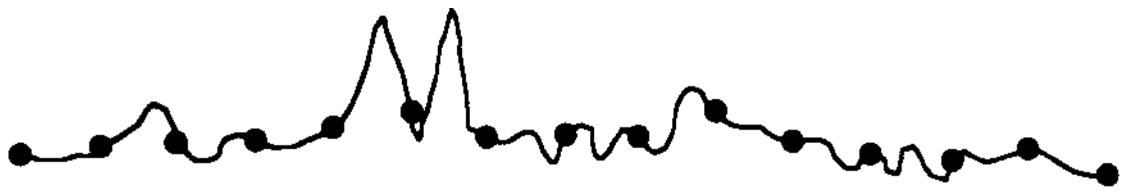
Figure 1: Fixed sampling. The input data are sampled once every fixed time interval.

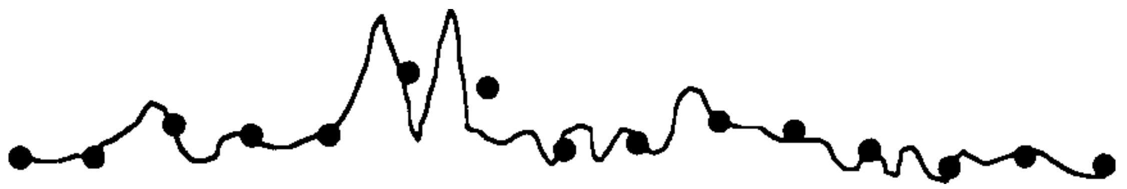
Figure 2: Averaged sampling. The input data are sampled once every fixed time interval. The value returned is the average of the parameter over the last sampling interval.

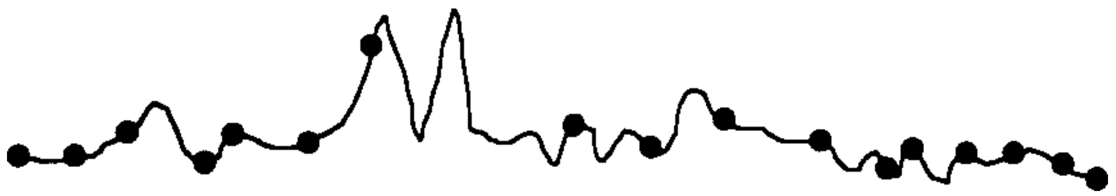
Figure 3: Polling. The input data are sampled whenever the system finishes processing the previous data.

should be monitored more closely. Conversely, irrelevant parameters should be monitored only closely enough to ensure that critical events are not lost.

These criteria can be met by using a combination of sampling and thresholding, with the sampling rate and thresholds dynamically controlled. In addition, the thresholds are made relative to the last data value sent. The sampling rate and threshold define a *dynamic filter* on a parameter.

The sampling rate is not a strict specification of the interval between incoming data values, but rather a *baseline* sampling rate: if no data values for a parameter have been sent within the time interval defined by the parameter's sampling rate, a new data value is sent. Thus a minimum sampling rate will be maintained. However the sampling rate may be higher, depending on the thresholds. The sampling rate is changeable by the system.

The real power of the approach comes from having a dynamic threshold set relative to the last data item received. This relative thresholding avoids the boundary conditions of the fixed thresholding approach. Since the allowable range within the threshold is changeable, the system alters the threshold to adjust the data rate.

Note that the use of thresholding as opposed to simple sampling provides information about the filtered data. More specifically, given a parameter's data value v and a threshold of $v \pm \Delta v$, the parameter is guaranteed to be in the range $[v - \Delta v, v + Av]$ until another data value is received.

The baseline sampling guarantees the data to be up to date, or no more out of date than the sampling interval. If the threshold is relatively small, then this will prove unnecessary. But with larger thresholds, this might prove important. Also, the working system uses this in its computation of data rates (this will be discussed in Section 4).

Dynamic filters allow the system to adjust the incoming data rate to its needs. Without any knowledge of its internal processing, the system could monitor the rate at which incoming data arrived in the system. If the rate rises too high or drops too low, the system could increase or decrease the filter threshold. This reactive adjustment acts like a control system with the function of keeping the rate within a given range. This approach is illustrated in Figure 5. Note that the rate of data sent to the system (the ones that the filters allow through) is independent of the rate at which data are sensed. Instead, the values of the sensed data determine the data rate to the system.

By inspecting the reasoning component, the system may notice a backlog of tasks waiting to be executed. The input could be slowed to allow the system to catch up. This needs to affect the input rate controller, which could otherwise negate any change that the task-level controller might make. Specifically, the task-level controller might slow the input rate down below the minimum allowable rate defined for the input-rate controller. The input-rate controller would then increase the data rate, possibly causing further backlogs on the task queue.

In addition to these reactive behaviors, it would be desirable to anticipate resource requirements and change the filters accordingly. The system might plan to execute a complex procedure, but before starting the procedure, it could change the filters to slow the data rate an amount appropriate for the complexity of the task. Note that this need only be approximate, since the components for rate control and task-backlog control will adjust the actual input rate to match the ongoing reasoning.

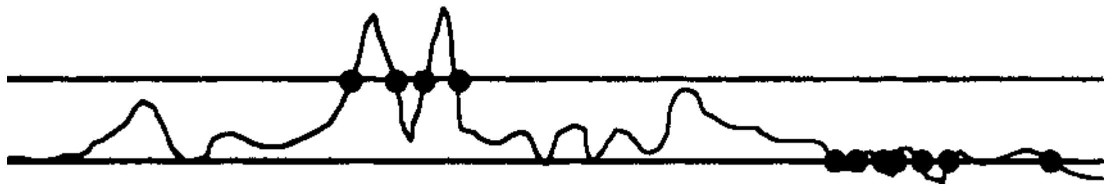
Figure 4: Fixed thresholding. The input data are sampled whenever they cross a fixed threshold.

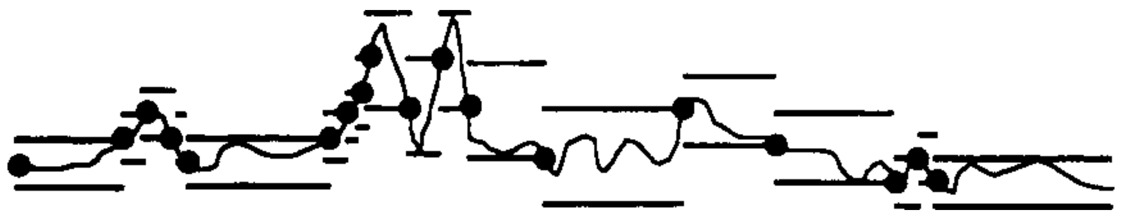
Figure 5: Dynamic filters. The input data are sampled when they cross a threshold around the previous data value sent. The size of the threshold varies according to the data rate.

Monitoring the task queue and anticipating resource requirements combine to satisfy the criterion that the system be responsive to reasoning resources. Since the system adjusts the filters in response to or in anticipation of reasoning, the incoming data rate remains at an appropriate level for the reasoning in the system.

The system still maintains its sensitivity to important data through the use of the dynamic thresholds of the filters. Each parameter is guaranteed to be within the range defined by its filter, and any deviation from that range will cause a new data value to be sent to the system.

To handle the criterion that attention be allocated differentially among the parameters, the system might have some notion of relevance. When a parameter is relevant to the reasoning task the system is performing, any changes to the filters would try to favor the relevant parameters over the irrelevant parameters. For instance, when a complex procedure begins, the system may be able to achieve a sufficient overall data-rate reduction by only changing the filters of parameters irrelevant to the procedure. The filters of relevant parameters would change only when changes to the irrelevant parameters were not enough.

Since the threshold ranges and the sampling rate are changeable, the system can control the amount of attention given a parameter. The system is then able to adjust the input rate for any parameter or set of parameters when it has knowledge that suggests that action.

4 Data Management in Guardian

These ideas for data management have been applied in the Backlog component of the Guardian system [Ilayes-Roth *et al.*, 1989]. Guardian is an application, implemented in the BB1 blackboard system [Hayes-Roth, 1985], for monitoring patients in a surgical intensive-care unit. Currently, Guardian monitors twenty parameters, performing tasks such as data abstraction, associative diagnosis, model-based explanation, and model-based diagnosis. These tasks vary in their complexity and consequently in their demands on reasoning resources. The Backlog subsystem of Guardian is responsible for maintaining appropriate data input levels.

There is a filter corresponding to each parameter that Guardian receives. These filters reside on an external processor, between the external sensors and Guardian. Guardian can change a filter by specifying the size of the relative threshold for a parameter (as in the amount of change allowable before a new data value is sent) or specifying the baseline sampling rate. In practice, the sampling rate stays constant while the threshold changes.

The maintenance of the incoming data rate is performed when data arrive in the system. Guardian has a number of input streams, each of which has an associated data rate. In the current system, there is one parameter per stream, but this is not required. The data rate is computed as a time-weighted average, decaying over time. Specifically, at the time a new data value is sent, if *At* is the time in seconds since the last data value was sent, and r is the input rate, in items per hour, at the time the last data value was sent, then

$$\left(\frac{1}{1+\Delta t}\right)r + \left(1 - \frac{1}{1+\Delta t}\right)\frac{3600}{\Delta t}$$

is the new data rate. This formula provides a reasonable balance between stability and responsiveness in the data input rate. Since the input rate is computed only when input arrives — this is to avoid loading the processor with continual recomputations — the data rate may start getting out of date, since in reality there is a continuous decay of the data rate when no input is arriving. The baseline sampling rate overcomes this problem by making sure data get sent at least once in the sampling interval. This ensures that the system's calculated data rate is no more than the sampling interval out of date.

When the data rate falls outside of the acceptable range, this condition is posted on the blackboard, which triggers a knowledge source for correcting the condition. The knowledge source has a priority proportional to the amount the data rate is out of range, so minor adjustments may be passed over when the system is busy. When the knowledge source runs, the filter thresholds for the associated parameters are changed an amount proportional to the amount the data rate is out of range. Specifically, if the maximum allowable data rate is r_0, the current data rate is $r > r_0$, the current threshold is Av, and the range of possible parameter values is *V*, then the formula

$$V\left[\frac{1-(\frac{r_0}{r})(1-2\frac{\Delta v}{V})}{2}\right]$$

defines the new threshold. When the data are distributed uniformly over the possible range of data, then $1-2\frac{\Delta v}{V}$ is the fraction of data actually being sent. Since the data rate is a factor of $\frac{r}{r_0}$ too high, the desired fraction of data sent is $\frac{r_0}{r}(1-2\frac{\Delta v}{V})$. The threshold formula is defined using this desired fraction. If the data are distributed uniformly, the formula will adjust the threshold by the correct amount to bring the data rate within the allowable range. In general it adjusts the data rate in the direction of the allowable range. A similar formula exists for the case where the data range is below the minimum allowable data rate.

The Backlog component also handles backlogs of pending tasks. The BB1 agenda is the queue for activated knowledge sources, and as such provides a ready indication of the number of outstanding tasks. When the agenda grows too large, a knowledge source is triggered, with a priority dependent on the size of the agenda. The knowledge source alters the input rate indirectly. Since, as mentioned earlier, the task-backlog correction needs to take precedence over the data-rate correction, the knowledge source triggered by the agenda overflow decreases the minimum and maximum allowable data rates for each stream by a percentage proportional to the size of the agenda. The data-rate adjustments will continue, except that they will now adjust the data rate to be within a lower range. This indirect control over the input rate meets the design goal for solving task backlogs. If the agenda continues to grow, the allowable ranges continue to decrease. When the agenda shrinks, the allowable ranges increase back towards their original settings.

Anticipation of reasoning-intensive tasks is done by monitoring the control plan of the blackboard. When a complex task is ready to run, the Backlog component adjusts the filters in proportion to the estimated complexity of the task. Currently this complexity is estimated ahead of time, but with more knowledge, the estimates could be computed at run time. As discussed earlier, the estimate need only be approximate, since the Backlog component will continue to monitor the data rate and the agenda, adjusting the rate to keep the incoming data at a reasonable pace for the task being performed.

To help focus the system's attention on important data, the system recognizes relevance of parameters to the reasoning tasks. Relevance is implemented by having objects on the blackboard linking parameters to plans (or more generally, parameters to an arbitrary blackboard object). This linking establishes the parameters as relevant in the context of the plan, and when the plan is active, changes to filters are made to other parameters first. The relevant parameters have their filters changed only when the changes to the other, irrelevant parameters result in less of an overall change than needed.

5 Performance

The Backlog component of the Guardian system has shown that it can meet the criteria for effective data management. The Guardian system has been run on scenarios containing up to three hours of simulated data, performing its various reasoning tasks while remaining abreast of the latest events in the data.

Although the large-scale changes in an intensive-care setting usually take place slowly, smaller variations may occur continually. In our current system, 15 parameters are sensed approximately every 20 seconds, with an additional 5 parameters — lab results and machine settings — sensed when they change. The duration of the BB1 reasoning cycle, averaged over a run of 45 minutes, is about 15 seconds. Since BB1 triggers at least one knowledge source for each data value it receives, and executes one per cycle, the filters must reduce the $(15/20)15 = 11.25$ data items sensed each cycle to one. This is a lower bound, and in reality, the number of sensed data versus the number of data processed is significantly higher. In a representative run of 45 minutes, each of 15 parameters was sensed 139 times, for a total of 2085 data values. The filters allowed between 4 and 22 of the 139 sensed values into BB1, with a total of 119 data values reaching the reasoning component. Thus the filters achieved greater than a 94% reduction in data, with the quality of the solution unaffected.

There is a delay between the system deciding to change the input rate and the change actually taking effect. At the least, there is some communication delay when the filtering component is on a separate machine (which is the case in the current system). In the case of agenda backlogs, the change is indirect — the system shifts the range of acceptable rates rather than changing the filters directly — so there may be a delay before any filter changes occur.

One possible problem with delays is when a sudden change occurs in the data variability. New filters are needed to preserve a reasonable rate of input to the system, but the filter change will be delayed, so either too much or too little data will arrive in the system. The current Guardian system has fixed-length buffers for input to the system, so if too much data arrives, the older data will be lost. When there is too little data, the system waits until the next data value arrives, and at that point adjusts the filters.

The effectiveness of the adjustment that the system makes to the filters is dependent on the delay in changing the filters. In particular, suppose that there were short bursts of highly varying data, interleaved with nearly flat data. If the bursts were close enough together, the system would start compensating for one extreme just when the other extreme occurred. This could be handled by noticing periodicity in the input data, but no such facility exists in the current system.

6 Ongoing Work

The current implementation of data management in the Guardian system includes the ideas presented in this paper. Further work is in progress to improve these ideas. In particular, a large part of the data management is being moved outside the reasoning component to avoid any unnecessary interference with the reasoning tasks, and the filter criteria are being expanded so that data may be filtered even more intelligently.

If a large portion of data management is done within the reasoning component of the system, it could potentially consume significant resources. This could make the overall performance of the system worse rather than better. In the current implementation of Guardian, the data management component is carefully constructed so that it will not interfere with the normal operation of the system. For instance, knowledge sources for data management are rated in proportion to the severity of the problem they are to correct. This way minor corrections will be put aside when important reasoning is underway.

A better approach to reduce the resource demands of data management is to move as much of the work as possible to another processor. Work is underway to move the data management task to the (remote) filtering machine, since a large part of its operation is independent of the reasoning. The parts that require information from the reasoning component are also being moved if they can get the information in a small amount of communication. For instance, the size of the agenda is easily communicated to a remote machine.

Another direction of work in progress is making the filtering criteria sensitive to more features. The reasoning system can set expectations, and the filtering component will use a violated expectation as another reason to send a data value, tagged appropriately. Also being added are classification ranges, so that when a data value changes from what the system considers *low* to *normal,* a data value will be sent. The filtering procedure allows these classification ranges to be set dynamically by the reasoning system. Additionally, some simple trend analysis will be performed to ensure that the overall rate of change in some parameter is within expected bounds. All of these additions are expected to be in place shortly, making the incoming data much more meaningful to the reasoning system.

7 Acknowledgements

This work was supported by grants from DARPA and NIH and gifts from Rockwell and FMC. Nicholas Parlante worked on an implementation of the external filtering program. Micheal Hewett helped on implementation and ideas in general. Luc Boureau is responsible for extending the ideas in the current system. Andrew Golding, Micheal Hewett, and Andrew Kosoresow provided helpful comments on earlier drafts.

References

[Ali and Scharnhorst, 1985] M. Ali and D. A. Scharnhorst. Sensor-based fault diagnosis in a flight expert system. In *Proceedings of the Second Conference on Artificial Intelligence Applications,* pages 49-54, 1985.

[Anderson *et al,* 1984] B. M. Anderson, N. L. Cramer, M. Lineberry, G. S. Lystad, and R. C. Stem. Intelligent automation of emergency procedures in advanced fighter aircraft. In *Proceedings of the First Conference on Artificial Intelligence Applications,* pages 496-501, 1984.

[Andersson, 1988] R. L. Andersson. *A Robot Ping-Pong Player: Experiment in Real-Time Control.* The MIT Press Series in Artificial Intelligence. The MIT Press, 1988.

[Chen, 1985] D. C. Chen. Progress in knowledge-based flight monitoring. In *Proceedings of the Second Conference on Artificial Intelligence Applications,* pages 441-446,1985.

[Fagan *et ai,* 1984] L. M. Fagan, J. C. Kunz, E. A. Feigenbaum, and J. J. Osborn. Extensions to the rule-based formalism for a monitoring task. In B. G. Buchanan and E. H. Shortliffe, editors, *Rule-Based Expert Systems,* chapter 22, pages 397-423. Addison-Wesley, 1984.

[Hayes-Roth *et al,* 1989] B. Hayes-Roth, R. Washington, R. Hewett, M. Hewett, and A. Seiver. Intelligent monitoring and control. In *Proceedings of the 1989 International Joint Conference on Artificial Intelligence,* 1989.

[Hayes-Roth, 1985] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence,* 26:251-321, 1985.

[Kaemmerer and Allard, 1987] W. F. Kaemmerer and J. R. Allard. An automated reasoning technique for providing moment-by-moment advice concerning the operation of a process. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87),* pages 809-813, 1987.

[Nitao and Parodi, 1985] J.J. Nitao and A. M. Parodi. An intelligent pilot for an autonomous vehicle system. In *Proceedings of the Second Conference on Artificial Intelligence Applications,* pages 176-183, 1985.