Reverse-Engineering Costs: How much will a Prognostic Algorithm save?

Chris Drummond and Chunsheng Yang
Institute for Information Technology
National Research Council Canada
Ottawa, Ontario, Canada, K1A 0R6
Email: (Chris.Drummond,Chunsheng.Yang)@nrc-cnrc.gc.ca

***Abstract*—One effective way to evaluate a prognostic algorithm is to estimate its potential cost savings. Unfortunately, the final cost is dependent on individual costs, seldom easy to obtain and changing over time. It is dependent on component failure rates, also subject to change. This paper shows a way of representing cost savings over a wide range of costs and failure rates. We show that, even without complete information, it is possible to "reverse engineer" the effective range of any algorithm. We introduce a simple tool to help the algorithm designer do this. We also argue that even when the information appears complete, it is useful to explore this range to assess the robustness of the algorithm. We illustrate the approach with a case study on prognostics applied to maintenance in the railway industry.**

I. Introduction

Evaluation of prognostic algorithms is an important exercise. It is necessary not only as a means of assessing the usefulness of a particular algorithm for a particular application but also to determine progress within the field as a whole. As the prognostics field expands rapidly, the need for good evaluation procedures becomes paramount. It is useful to draw from closely related fields, such as Machine Learning. Experience, there, has shown that the over reliance on a single performance measure, such as error rate, is problematical. This paper discusses a graphical representation which can cope more effectively, than any single measure, with the changes that naturally occur within domains such as equipment maintenance. We do not claim that this approach can account for all possible variations that should be taken into account. However, modeling performance in terms of cost savings, how much a prognostic algorithm improves on the current approach, is very intuitive and readily understood by end users.

One difficulty, in obtaining a final estimate, is that cost savings depend on individual costs not always easy to obtain. This may occur because no-one has all the information readily at hand. It may be due to people's reluctance to count the costs associated with safety issues. In this paper, we explore the idea of reverse engineering some of these values during the development process. What we mean by reverse engineering is determining the range of costs and failure rates over which an algorithm will be useful. This range may be sizable, perhaps up to one or two orders of magnitude in cost. So, even without a good estimate of actual costs, we would have some confidence that the algorithm will be useful in practice. Presenting a range of values should also make the end user more comfortable. Users are reluctant to give single values themselves, with some justification as sizable cost variances are inherent in the domain. We do not expect reverse engineering of these values to occur just once. We view this as part of a continuous dialog between the algorithm designer and end user.

Even if we do have estimates of costs from the end user, we should treat them as "best guesses" rather than exact values. Therefore, establishing a range is still useful. It it also extremely unlikely that the values are static. Over time, many things will change. How often a part fails will change. The maintenance organization and original equipment manufacturer will address problems extending a part's life. Other aspects, such as the conditions to which the equipment is exposed, will also affect failure rates. How much the part costs will change. Parts will fluctuate in price, due to improved manufacturing or other changes outside the maintenance organization's control. The cost of repairs will change. The maintenance organization will work to streamline its procedures to reduce repair time and costs, yet labor costs may increase. The consequence of a part failing will change. Without some idea of the robustness of a prognostic algorithm to changes, a single performance value would be of questionable merit.

Here, we introduce a tool to help algorithm designers evaluate their performance in comparison with what the end user is currently doing. We have used this process for algorithms we are developing ourselves. We apply it to maintenance in the railway industry to illustrate its advantages.

II. Visualizing Performance

This section introduces a way of visualizing the expected cost of different maintenance policies. The approach is a specialization of a general method for visualizing classifier performance called cost curves [1], [2]. To introduce this visualization method, let us begin by imagining that failures are very rare. This is the lower left hand corner of figure 1. Suppose we wait for a failure to occur before doing anything. Let us call this trivial policy "Replace on Failure". This is a good policy when failures are rare. However, as failures become more common, the solid arrow at the bottom, the overall cost will increase, the solid arrow on the left. The relationship between the two is indicated by the dashed arrow.

The cost is directly proportional to the probability of failure, the expected cost is just the cost of a failure times its

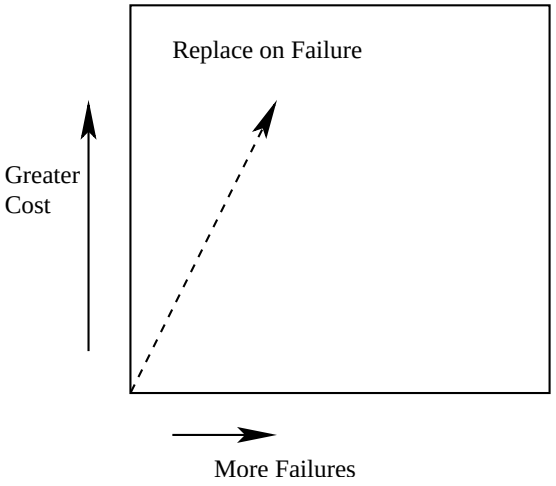

Fig. 1. Increasing Costs

probability. If the failure rate is constant but the cost of failure increases we would have the same result. So, changes in the probability of failure and the cost of failure have similar effect. If we normalize the product of probability and cost so that it ranges from zero to one, we end up with the x-axis of figure 2. As we have normalized the x-axis, the y-axis, the expected cost also ranges from 0 to 1. One is the maximum cost that could occur. The closer the performance is to zero on the y-axis the lower the expected cost.

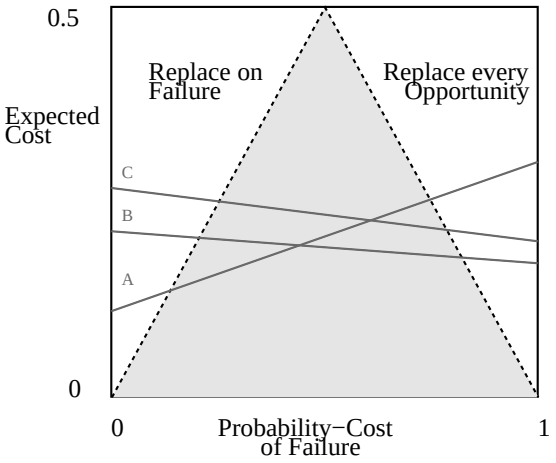

Fig. 2. The Operating Region

If the failures become too common, or the cost of failures too high, rather that wait until a part fails we would be better using the trivial “Replace every Opportunity” policy, as shown in figure 2. (We assume we can replace parts fast enough to avoid failure). In practice, maintaining equipment under these circumstances is probably futile. It does however establish a clear region, indicated by the light gray triangle, where useful prognostic algorithms must operate. If the curve representing the performance of the algorithm strays outside this triangle, it is not useful for that range of probability-cost values. We call the useful range of x values the algorithm’s operating range.

We can also compare the performance of different prognostic algorithms against each other and the maintenance policy used by the organization. Thus we can easily visualize the cost savings obtained when using a prognostic algorithm instead of an existing policy. The straight dark gray lines in figure 2 show the performance of three prognostic models. For each model, the likelihoods are fixed. So, the expected cost will also be a straight line, the linear sum of likelihoods times costs times failure rate [1]. Looking at the lines labeled A and B we can see that each achieves lower expected cost than the other for some range costs and failure rate. The vertical distance between the lines represents the cost savings. Noticeably, the line labeled C is not best for any range, so will not be useful under any circumstances. Both lines A and B leave the light gray triangle, defining the limits of their operating ranges. A prognostic algorithm will produce many such models and therefore many lines. These can be combined to form a curve showing the expected performance of the algorithm over the full range of costs and failures rate.

One problem with this representation is its very generality. It shows the expected cost for each algorithm, but the values are normalized between zero and one. Under normalization, the ratio of costs is maintained, important when comparing different algorithms. However, it does not show actual dollar figures, valuable to the end user. Here, we both simplify and extend the cost curve representation to allow such figures to be determined, say to easily establish the dollar range where a prognostic algorithm is useful. We call the extension reverse engineering, because, with only approximate knowledge, we can reverse engineer both the potential cost savings and the effective range of the algorithm. Figure 3 shows the additional interface. The figure shows boxes for two costs: the cost of a false alarm and the cost of an undetected failure.

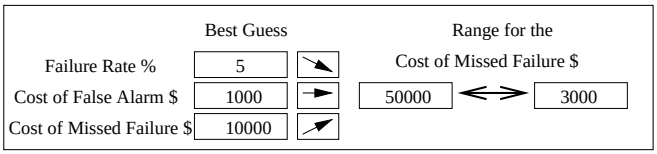

Fig. 3. Reverse Engineering Costs

By entering ”best guess” cost and failure rate information, we can extract the range in terms of either of the two costs or the failure rate by clicking the ”arrow button” beside each item (the entered value is essentially the “mean”). Often, the failure rate and cost of a false alarm are relatively easy to estimate. The failure rate may be a matter of historical record. The maintenance actions needed to determine if a fault has actually occurred may be specified. The cost of these actions, in terms of time spent and material used, may be recorded as part of standard operation. The cost of an undetected failure, however, may be more problematic as there are costs outside the maintenance actions. So, for example, in Figure 3, when we click the ”arrow button” aligned with the cost of missed failure box, its range will be shown on the right side of the interface based on the given failure rate and cost of false alarm.

III. A Case Study

This section shows how our method works on a real-world application, predicting problems with train wheels [3].

A. The Application Domain

Train wheel failures cause half of all train derailments and cost the rail industry billions of dollars a year. They also accelerate rail deterioration, ultimately leading to broken rails. These breaks are dangerous and very expensive to repair. The risk of wheel failure is increasing as global competitiveness pushes railways to use larger and heavier cars. To limit failures, wheels are closely monitored using impact detectors [4]. Installed at strategic locations on the rail network, these measure the dynamic impact of each wheel. When the impact exceeds a threshold, 140 kips [1], the train must immediately reduce speed and stop at the nearest siding. The offending wheel must be replaced before the associated car is used again. Although this reduces some of the costs associated with wheel failure, such as those due to derailment and rail damage, it increases other costs. Replacing a wheel can be a costly event, particularly if the siding is remotely situated.

To reduce these extra costs, the Association of American Railroads established policies [5] to help railways make maintenance decisions. An impact below 140 kips can be used for prediction. In one policy, whenever the impact of a wheel exceeds 90 kips a message is sent to the train operator. The operator can use this to schedule maintenance on the affected car but, due in part to the very high frequency of these alarms, they are often ignored. Our own study suggests that operators have good reasons for this. First, a large number of 90 kips events never lead to wheel failure. There are a large number of false alerts, and many good wheels replaced. Second, when wheel failure did occur, the time from the warning to a failure varies largely. It is difficulty to schedule accurately when maintenance should occur.

B. Cost saving estimation for prognostic algorithms

To address these limitations, we began the WILDMiner project [3]. Our data mining algorithms learn models from impact data to produce real-time prognostics. For this study, we used data collected over 17 months from a fleet of 804 large cars with 12 axles each. This produced a data set containing 2,409,696 impact measurements grouped in 9906 time-series. We used 6400 time-series for training (roughly the first 11 months) and kept the remaining 3506 time-series for testing (roughly the last 6 months). Since there are only 129 occurrences of wheel failures in the training data set, we selected the corresponding 129 time-series out of the initial 6400 time-series in the training data set. We created a data set for modeling which contains 214364 measurements from the selected 129 time-series. We built prognostic models following our own data mining methodology [6]. Figure 4 shows the performance of the overall algorithm, details can be found elsewhere [7]. We ran experiments using 13 different cost matrices, different values for the cost of a false alarm and an undetected failure, linearly interpolated to give the continuous red line. The green line is for the 90 kips threshold. The line is straight as the model is unchanged by a variation in cost or failure rate, but the expected cost saving does change.

Fig. 4. Performance of a prognostic algorithm

The interface is used to determine where and by how much the prognostic algorithm improves over the existing approach. It also shows how they both fare in comparison with the trivial policies of "replace on failure" and "replace every opportunity". The present, 90 kips, policy is generally better than waiting until the wheel breaks, the left hand side of the figure. But it would be completely the wrong policy should the failure rate, or cost of a failure, become very high, on the right hand side. Across most of the range our prognostic algorithm is clearly much better. It offers significant savings over the present policy. Pressing the limits button, at the top, produces a vertical blue dashed line, which can be dragged to anywhere in the figure. A middle mouse click on this line shows the expected cost values at that point. With the line in the position shown, the present policy represents a cost saving of a bit less than 20% (1-0.41/0.50), our prognostic algorithm a cost saving of just over 50% (1-0.27/0.50). The maximum saving occurs slightly to the right of center of the figure. Here, the present policy offers no saving, our prognostic algorithm a saving of about 56%. Unlike the 90 kips policy, our algorithm would still be useful for high failure rates and costs, so is considerably more robust.

The actual savings will depend on the costs and failure rate that occurs in practice. If we don't know these values, then the best we can do is to establish a range over which some savings are achieved. In our tool, by pressing the limits button a second time, we add the vertical dashed red lines in Figure 4. We then drag them to where the prognostic algorithm intersects the existing policy or to the ends of its operating range, which ever is the lower. We could also position them closer together so that the range always includes some worthwhile cost savings. Noticeably, in the lower left hand corner all the policies have around the same cost. Unless we can improve our prognostics

[1] A kip is equal to 1,000 pounds. A 140 kips wheel impact is a combined static and dynamic force of 140,000 pounds exerted by an individual wheel at the wheel/rail interface when the freight car passes over the WILD site.

algorithm, waiting until failure occurs is as good as doing anything else. The good news is that our algorithm is effective over a wide range of probabilities and costs. The range is greater than 30:1, more than an order of magnitude.

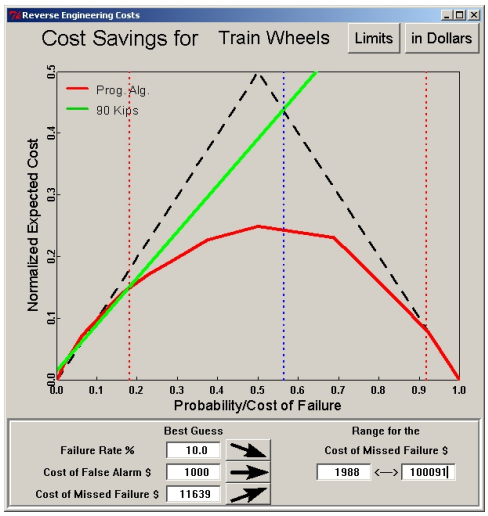

Fig. 5. Reverse Engineering Costs

The main costs in this application consist of that for a false alarm (perhaps simply a visual inspection of the wheel) and that for a missed failure (a wheel failure during normal operation). We would like to express the useful range of the prognostic algorithm in terms of these values. We expect that these would be more readily understood by the end user. By pressing the button marked "as a ratio", the interface is extended with a series of boxes. To do this we need some idea of typical costs. We begin by putting in our best guess for the failure rate say 1 in 10 or 10%, and the cost for a for a false alarm, say $1000. We can then explore the range of possible costs for a missed failure for which our algorithm would be useful. Figure 5 shows this example. The maximum cost savings would be achieved for a missing failure cost of about $12000 and the range would be about $2000 to $100,000.

We argue that establishing a range is critical even when costs are more trustworthy. To evaluate prognostic models, Yang and Létourneau [7] developed both a score-based and a cost-based method. The cost-based approach required complete and accurate cost information. We used information provided by an independent expert in the railway industry: $500 per false alert, $5000 per undetected failure. How certain can we be that these values are correct? The first should be easy to obtain directly from maintenance records. The second is much more difficult to estimate. A failure may cause a secondary component to fail prematurely. It may cause delays in the scheduled operations with negative consequences on user satisfaction and the company's reputation. It may cause a safety hazard or even a catastrophe, costing millions of dollars and loss of life. These secondary effects could greatly influence decision regarding the adoption of a prognostic model. With a good range of values, we can have some

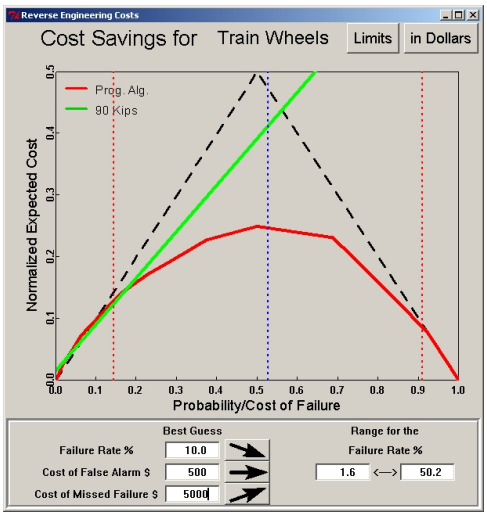

Fig. 6. Using Expert supplied Values

confidence we can deal with unforeseen costs. In Figure 6, we used a failure rate of 10%, the value in our data set. With these values we get almost the maximum cost savings and we have a large useful range indicated by the two red dashed lines in the figure. By pressing the buttons marked with arrows, we can the range in terms of the failure rate or the two costs. The figure shows the range for the failure rate (about 2% to 50%). In terms of the cost of missed failure, the range is about $750 to $45,000; in terms of the cost of false alarms the range is about $55 to $3500.

IV. Conclusions

This paper has introduced a way to evaluate a prognostic algorithm by estimating the cost savings that will be achieved if it is deployed. We introduced a simple tool to help the algorithm designer do this. We show that, even without accurate or even complete information, it is possible to "reverse engineer" the effective range of any algorithm. This should give the designer some confidence the system will be useful in the face on unseen or changing costs.

References

[1] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," *Machine Learning*, vol. 65, no. 1, pp. 95–130, 2006.

[2] C. Drummond, "Changing failure rates, changing costs: Choosing the right maintenance policy," in *Proc. of AAAI Fall Symposium on Artificial Intelligence for Prognostics*, 2007.

[3] C. Yang and S. Létourneau, "Learning to predict train wheel failures," in *Proc. of 11th international conference on Knowledge discovery in data mining*, 2005, pp. 516–525.

[4] S. Lechowicz and C. Hunt, "Monitoring and managing wheel condition and loading," in *Proc. of International Symposium on Transportation Recorders*, 1999, pp. 205–239.

[5] A. A. Railroads, "Advanced technology safety initiative: Equipment health management system," *Association of American Railroads*, 2004.

[6] S. Létourneau, C. Yang, C. Drummond, E. Scarlett, J. Valdés, and M. Zaluski, "A domain independent data mining methodology for prognostics," in *Proc. of Essential Technologies for Successful Prognostics*, 2005.

[7] C. Yang and S. Létourneau, "Model evaluation for prognostics: Estimating cost saving for the end users," in *Proc. of 6th International Conference on Machine Learning and Applications*, 2007, pp. 304–309.