Multiobjective Evolution of Temporal Rules

Pål Sætrom[1] and Magnus Lie Hetland[2]
[1]*Interagon AS, Medisinsk-teknisk senter,*
NO–7489 Trondheim, Norway
paalsat@interagon.com
[2]*Norwegian University of Science and Technology,*
Dept. of Computer and Information Science,
Sem Sælands vei 9, NO–7491 Trondheim, Norway
magnus@hetland.org

Abstract. In recent years, the methods of evolutionary computation have proven themselves useful in the area of data mining. For rule mining, several objective functions have been used, relating to both accuracy and interestingness in general. However, when searching for rules or patterns in a data set, several conflicting objectives will often be present. As the ultimate goal of data mining is to discover unexpected, useful knowledge, it may not be feasible to prioritize these objectives *a priori*. Simply constructing an aggregate fitness function in these cases could be seen as a more or less *ad hoc* solution. In this paper we propose an alternative: Using well-established multiobjective evolutionary algorithms to evolve a Pareto optimal set of rules.

1 Introduction

In recent years, the methods of evolutionary computation have proven themselves useful in the area of data mining. For the problem of rule mining, several objective functions have been used, relating to both accuracy and interestingness in general [1, 2]. However, when searching for rules or patterns in a data set, several conflicting objectives will often be present. Such objectives might include measures of accuracy and interestingness, as well as readability or parsimony. As the ultimate goal of data mining is to discover unexpected, useful knowledge, it may not be feasible to prioritize these objectives *a priori*. Simply constructing an aggregate fitness function in these cases could be seen as a more or less *ad hoc* solution. In this paper we propose an alternative: Using well-established multiobjective evolutionary algorithms. These produce an approximation of the Pareto front in the multiobjective search space. An expert user may then inspect the resulting rule set to decide which rules are of potential use. We demonstrate the idea using the SPEA2 algorithm [3], combined with the temporal rule mining approach described in [2].

2 Method

In [2] a method for doing unsupervised data mining in time series is presented. The method is based on genetic programming and generates rules from a prespecified rule language by optimizing a function that measures the (perceived) interestingness of a rule. This is an aggregate function, which combines several aspects of rule quality into a single measure.

In the following sections, a multi objective genetic programming (MOGP) algorithm based on the ideas of [2] and [3] will be outlined. In Sect. 2.1 we give a brief description of our time series preprocessing. Following that, in Sect. 2.2 the rule format and internal rule representation used by the algorithm is described. Section 2.3 gives a brief outline of multiobjective optimization and the SPEA2 algorithm. In addition, some small modifications in the SPEA2 algorithm, used in the MOGP algorithm, are presented. Section 2.4 describes the objective functions used by the MOGP algorithm. Finally, Sect. 2.5 briefly outlines how the objective functions are evaluated.

2.1 Discretization

The time series data are discretized by sequentially extracting a real-valued feature from a sliding window, in this case the slope of a line fitted to the points in the window through linear regression. Following this feature extraction, discretization limits are found for a set of symbols in an alphabet Σ (see Sect. 2.2) in a manner that ensures a uniform distribution of the symbols in the resulting data set. For more information about the discretization process, see [2].

2.2 Rule Representation

The basic rule format that will be used throughout this paper is the simple and well known: "If *antecedent* then *consequent* within T time units." In general, the rule format can be formalized by defining the respective languages L_a and L_c that the antecedent and consequent can belong to. Several different languages have been used in the literature, ranging from single symbols from a fixed alphabet Σ [4] to relatively complex pattern languages [2].

The mining algorithm works by using genetic programming to search the space of possible rules defined by L_a, L_c and T. More specifically, each individual in the population is a syntax tree in the language $L_a \overset{T}{\Rightarrow} L_c$. This is implemented by using three separate branches; One branch for each of L_a, L_c, and T.

In the antecedent and consequent branches, the internal nodes in the parse tree are the syntactical nodes necessary for representing expressions in the corresponding languages. If for example, the considered language is regular expressions, the syntactical nodes needed are *union*, *concatenation* and *Kleene closure*. The leaf nodes in these branches are the symbols from the antecedent and consequent alphabets (Σ_a and Σ_c).

The function of the maximum distance branch, T, is to set the maximum distance of the rule. Hence, the branch is constructed by using arithmetic functions (typically $+$ and $-$) as internal nodes, and random integer constants as leaf nodes. The final distance is found by computing the result of the arithmetic expression r_T, and using the residue of $r_T \bmod T+1$.

2.3 Multiobjective Evolution

Multiobjective optimization is the problem of simultaneously optimizing a set F of two or more objective functions. The objective functions typically measure or describe different features of a desired solution. Often these objectives are conflicting in that there is no single solution that simultaneously optimize all functions. Instead one has a *set* of optimal solu-

tions. This set can be defined using the notion of *Pareto optimality* and is commonly referred to as the *Pareto optimal set* [5].

Assuming that the functions in F should be maximized, then a solution $\mathbf{x}$ is *Pareto optimal* if there no other solution $\mathbf{x}'$ exists such that $f_i(\mathbf{x}') \geq f_j(\mathbf{x})$ for all $f \in F$ and $f_i(\mathbf{x}') > f_j(\mathbf{x})$ for at least one $f \in F$. Informally, this means that $\mathbf{x}$ is Pareto optimal if and only if there does not exist a feasible solution $\mathbf{x}'$ which would increase some objective function without simultaneously decreasing at least one other objective function.

The solutions in the Pareto optimal set are called *non-dominated*. Given 2 solutions, $\mathbf{x}'$ and $\mathbf{x}$, $\mathbf{x}'$ *dominates* $\mathbf{x}$ if $f_i(\mathbf{x}') \geq f_j(\mathbf{x})$ for all $f \in F$ and $f_i(\mathbf{x}') > f_j(\mathbf{x})$ for at least one $f \in F$. In other words, $\mathbf{x}'$ is at least as good as $\mathbf{x}$ with respect to all objectives and better than $\mathbf{x}$ with respect to at least one objective.

The goal in multiobjective optimization is to find a diverse set of Pareto optimal solutions. In evolutionary multiobjective optimization this is typically found by producing a set of solutions from a single evolutionary algorithm run. Several different algorithms for evolutionary multiobjective optimization exist (see [5] for an introduction and [6] for a survey).

The algorithm used here is based on the SPEA2 [3], which uses a fixed size population and archive. The population forms the current base of possible solutions, while the archive contains the current solutions. The archive is constructed and updated by copying all non-dominated individuals in both archive and population into a temporary archive. If the size of this temporary archive differs from the desired archive size, individuals are either removed or added as necessary. Individuals are added by selecting the best dominated individuals, while the removal process uses a heuristic clustering routine in objective space. The motivation for this is that one would like to try to ensure that the archive contents represent distinct parts of the objective space. The fitness of an individual is based on both the strength of its dominators (if dominated) and the distance to its k-nearest neighbor (in objective space). See [3] for further details.

In this work the SPEA2 algorithm has been modified as follows. When selecting individuals for participation in the next generation, both the archive and the main population were used. The SPEA2 approach of only selecting from the archive was tried, but this resulted in premature convergence, and the results in the final generation were simple variations of the first archive contents. In addition, to prevent further convergence of the archive contents, only individuals having differing objective values were selected in the initial archive filling procedure. If two or more individuals shared the same objective values, one of these was randomly selected to participate in the archive.

In our experiments, the population size was typically 100 times larger than the archive size. Subtree swapping crossover was used 99% of the time, while tree generating mutation was used 1% of the time.

2.4 Objective Functions

Rules generated by an automatic data mining algorithm should often satisfy several requirements. For example, the rules should be accurate, interesting and comprehensible [1]. In the following, formalizations of these notions in the form of real-valued functions are presented. These formalisms are then used as objective measures in the multiobjective evolution.

2.4.1 Accuracy

Given a rule $R = R_a \overset{t}{\Rightarrow} R_c$ in the rule language $L_a \overset{T}{\Rightarrow} L_c$ (such that $t \leq T$ – see Sect. 2.2), and a discretized sequence $S = (a_1, a_2, \ldots, a_n)$, the frequency $F_S(R_a)$ of the antecedent is the number of occurrences of R_a in S. This can be formalized as

$$F_S(R_a) = |\{i \mid H(R_a, S, i)\}|, \tag{1}$$

where $H(R_a, S, i)$ is a hit predicate, which is true if R_a occurs at position i in S and false otherwise. The relative frequency, $f_S(R_a)$, is simply $F_S(R_a)/n$, where n is the length of S.

The *support* of a rule is defined as:

$$F_S(R_a, R_c, t) = |\{i \mid H(R_a, S, i) \wedge H(R_c, S, j) \wedge i{+}1 \leq j \leq i{+}t\}| \tag{2}$$

This is the number of matches of R_a that are followed by at least one match of R_c within t time units.

The *confidence* of a rule is defined as:

$$c_S(R) = \frac{F_S(R_a, R_c, t)}{F_S(R_a)} \tag{3}$$

The confidence measures the accuracy of the antecedent at predicting the consequent, while the support gives a measure of how well the rule represents the data. A rule having very low support, typically reflects freak incidents or noise in the data and thus is neither particularly accurate nor interesting.

2.4.2 Interestingness

The term interestingness is one commonly used in the field of data mining to denote the degree of surprise associated with the discovery of a rule. Several different interestingness measures have been developed (see [7] for a survey). The J-measure ([8]), is one particular measure which has already been proven useful in mining time series. This is defined as

$$J(R_c^t, R_a) = p(R_a) \cdot \Big(p(R_c^t|R_a) \log_2 \frac{p(R_c^t|R_a)}{p(R_c^t)} + (1 - p(R_c^t|R_a)) \log_2 \frac{1 - p(R_c^t|R_a)}{1 - p(R_c^t)} \Big). \tag{4}$$

Here, $p(R_a)$ is the probability of $H(R_a, S, i)$ being true at a random location i in S. $p(R_c^t)$ is the probability of $H(R_c, S, i)$ being true for at least one index i in a randomly chosen window of width t. Finally, $p(R_c^t|R_a)$ is the probability of $H(R_c, S, i)$ being true at for at least one index i in a randomly chosen window of width t, given that $H(R_a, S, j)$ is true and that j is the position immediately before the chosen window. The J-measure combines a bias toward more frequently occurring rules (the first term, $p(R_a)$), with the degree of surprise in going from a prior probability $p(R_c^t)$ to a posterior probability $p(R_c^t|R_a)$ (the second term, also known as the cross-entropy).

2.4.3 Comprehensibility

One of the most important principles of mining comprehensible rules is using a rule representation that in itself is intelligible. In addition, one often tries to limit the size of the rules. This is motivated by the fact that larger rules usually are harder to interpret. When using genetic

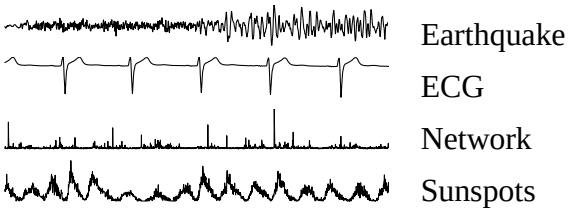

Figure 1: The time series analyzed.

programming (GP) as the rule induction method this becomes even more important. This is because GP tends to create large programs that contain semantically irrelevant parts. This tendency toward large programs is known as *bloat*.

Equation (5) gives a definition of *rule complexity*, which is used in the following experiments.

$$complexity(R) = (nodeCount(R) + maxDepth(R))^{-1} \tag{5}$$

Here the functions $nodeCount(R)$ and $maxDepth(R)$ return the number of nodes and the maximum depth of R, respectively.

2.5 Rule Evaluation

As in [2], a special purpose search chip [9, 10] is used for finding the support and confidence of each rule. It is also used for estimating the probabilities needed for calculating the J-measure. Spurious correlations introduced by the discretization process are circumvented by setting a minimum distance d_w between the antecedent and consequent in all rules generated for a sequence discretized with window size w. The comprehensibility is computed by a simple traversal of each branch in the rule tree (see Sec. 2.2).

3 Experiments

The MOGP algorithm was tested on four data sets from the UCR Time Series Data Mining Archive [11]: ECG measurements from several subjects, concatenated; Earthquake-related seismic data; Monthly mean sunspot numbers from 1749 until 1990; and Network traffic as measured by packet round trip time delays. Figure 1 shows plots of sub-sequences of the different time series analyzed.

We performed four sets of experiments. First, the four time series were mined using the four objective functions from section 2.4: support (2), confidence (3), J-measure (4), and rule complexity (5). Second, the time series were again mined, but now only the confidence, J-measure, and rule complexity were optimized. This was motivated by the fact that the J-measure implicitly rewards frequently occurring rules via the $p(R_a)$ factor (see (4)). Thus, in theory, there should be no need to optimize the support explicitly. Third, the MOGP algorithm was modified so that the final rule set would contain rules having differing consequents and the time series were again mined. Fourth, we performed a set of experiments to determine how the window size in the discretization algorithm influenced the results of the mining algorithm.

The following sections outline the results of the four sets of experiments. All results were generated by running the MOGP algorithm with a population size of 1000 and archive size of 10 for a maximum of 100 generations. We used the IQL language [12] as a template for

Table 1: Typical archive contents at algorithm termination for the ECG dataset

Rule	*J*-mea.	Conf.	Supp.	Compl.
$t \stackrel{5}{\Rightarrow} s$	0.057	0.67	0.033	0.20
$b \stackrel{9}{\Rightarrow} c$	0.050	0.75	0.038	0.20
$t \stackrel{6}{\Rightarrow} s$	0.053	0.67	0.033	0.20
$!(\geq \text{rnokoussrfisehznh}) \stackrel{9}{\Rightarrow} g$	0.020	0.48	0.41	0.020
$!((\geq o)(\geq \text{nokoussrfisehznh})) \stackrel{9}{\Rightarrow} g$	0.037	0.52	0.39	0.019
$\leq \text{rnokoussrfisehznhdgv} \stackrel{9}{\Rightarrow} g$	0.021	0.48	0.41	0.017
$\leq \text{rnokoussrfisehznhv} \stackrel{9}{\Rightarrow} g$	0.020	0.48	0.41	0.019
$\leq \text{rrnouonequsehznhv} \stackrel{9}{\Rightarrow} g$	0.017	0.47	0.41	0.020
$\leq \text{rnoononequsehznhv} \stackrel{9}{\Rightarrow} g$	0.020	0.48	0.41	0.020
$\leq \text{rnouonequssrfisehznhv} \stackrel{9}{\Rightarrow} g$	0.021	0.48	0.41	0.017

generating rules. The antecedents were IQL expressions, while the consequents were single characters or concatenations thereof.

3.1 Using Support, Confidence, J-measure, and Rule Complexity

Table 1 lists the results from a run on the ECG data set discretized with a window size of 2. The hits of the first, second and fifth rules in a subsequence of the ECG series are plotted in Figure 2.

Table 1 and Figure 2 are representative of the solutions obtained on the ECG set. Two observations can be made from these results. First, the archive apparently has converged, as many of the rules are simply minor variations of other rules. Second, the results illustrate an effect observed in runs on the other data sets. The archive typically contains two groups of rules: One group with rules having *confidence* $\gg$ *support*, and another group having *confidence* < 0.5 and *confidence* $\approx$ *support*. As the plot of the fifth rule from Table 1 in Figure 2 shows, the antecedents in the latter group typically match almost every position in the sequence. Because of this, these rules are of little or no value to a human user.[1]

To conclude this section, Figure 3 presents some of the rules mined from the different time series. As these plots show, the MOGP algorithm was able to generate rules that recognize and predict the significant features of the different time series. These include a rule for recognizing the increased oscillations occurring during an earthquake, rules that recognize the major peaks in the sunspot and ECG data, and a rule that is partly able to detect the periods of increased network activity.

3.2 Using Confidence, J-measure, and Rule Complexity

Table 2 lists a subset of the results of the reanalysis of the ECG data discretized with window size 2. In addition, the archive contained 3 versions of the first rule having differing maximum

[1] Note, however, that the positions where the antecedent of the fifth rule in Table 1 does *not* match correspond to the peaks of the ECG sequence. In other words, the inverse of an antecedent may also reveal interesting aspects of a sequence.

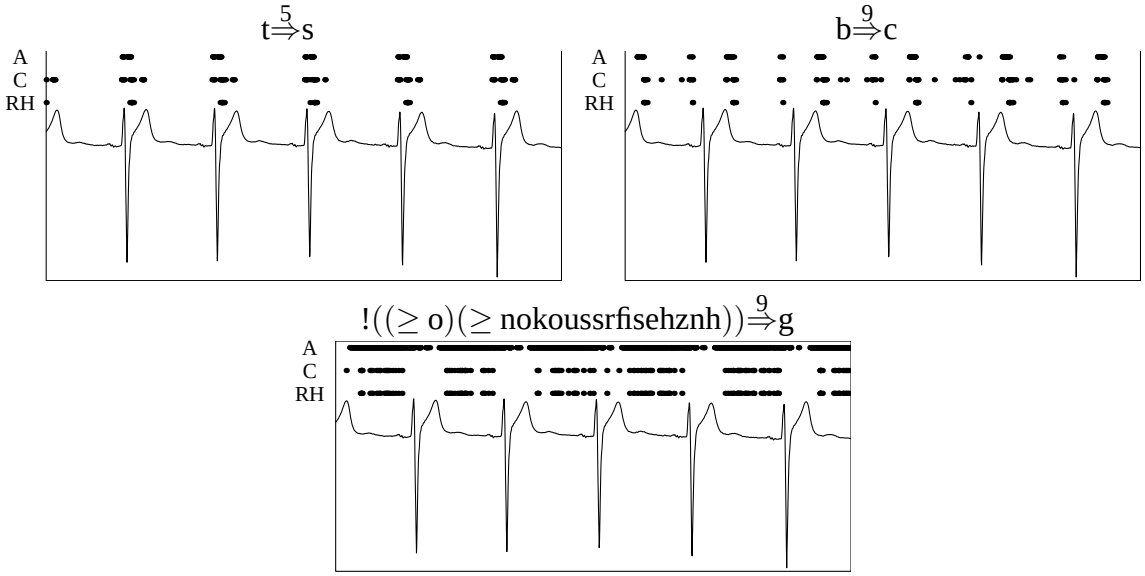

Figure 2: Hit locations in a subsequence of the ECG series of selected rules from Table 1. The dots following the *A* and *C* labels on the *y*-axis indicate the hit locations of the antecedent and consequent, respectively. The dots following the *RH* label indicate the positions where the rule hits, that is where the consequent follows the antecedent within the desired distance.

Table 2: Selected archive contents at algorithm termination for the ECG dataset when not optimizing the support

Rule	*J*-mea.	Conf.	Supp.	Compl.
$b \overset{2}{\Rightarrow} c$	0.058	0.58	0.029	0.20
$qq(\geq n \geq n \geq n)q \overset{4}{\Rightarrow} r$	0.017	0.92	0.0063	0.042
$\geq n(t \overset{49}{\longleftarrow} (\geq q \overset{83}{\longleftarrow} \geq n \geq c \geq n \geq n)) \geq c \mid$ $\geq c(t \overset{49}{\longleftarrow} (\geq q \overset{83}{\longleftarrow} \geq n \geq c \geq n \geq n)) \geq n \overset{9}{\Rightarrow} q$	0.18	0.77	0.13	0.028
$qb \mid bq \overset{1}{\Rightarrow} c$	0.000059	1.0	0.000014	0.13

distance, and 3 versions of the third rule having small variations in the antecedent. The hits of the rules from Table 2 in a subsequence of the ECG series are plotted in Figure 4.

As can be seen, removing the explicit support optimization did not adversely affect the support of the generated rules. Some of the generated rules did have very low support and *J*-measure values (typically corresponding to a single occurrence of the rule in the data set). Rules like these were, however, also generated when the support was optimized directly (data not shown).

3.3 Promoting Differing Consequents

To further stimulate the discovery of rules exploring different properties of the time series, the domination relation in the MOGP was modified slightly: One rule could only dominate another rule if both rules shared the same consequent. In addition, all rules with no support were automatically dominated.

This approach did not however have the intended effect. Even though the resulting rules

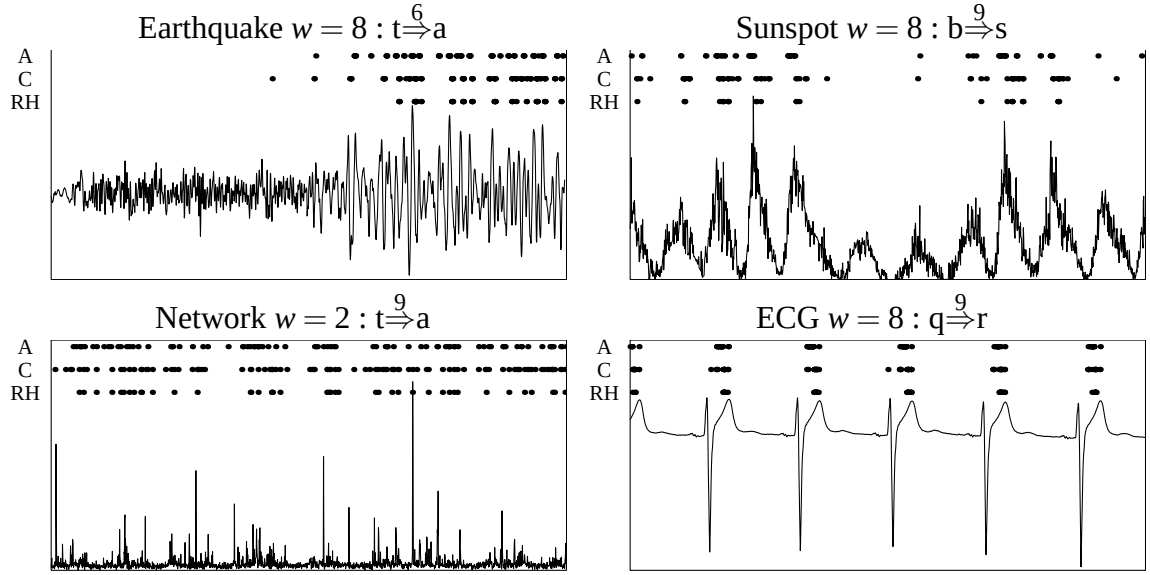

Figure 3: Hit locations of selected results on different sequences (see Figure 2 for the definitions of *A*, *C*, and *RH*).

did have differing consequents, most of the rules produced were either highly specialized (having confidence ≈ 1 and low support and J-measure), or had very low confidence (data not shown). Thus it seemed that instead of the multiple almost identical rules produced earlier, the system now produced a few relevant rules and several unwanted and uninteresting rules. These results suggest that the MOGP algorithm can find rules involving the most interesting consequents, without using the modification described in this section. As a result, this approach was abandoned.

3.4 Investigating the Window Size Effect

We performed several analyses of the four time series discretized with different window sizes. This revealed that by increasing the window size, the MOGP algorithm was better able to generate rules for recognizing the characteristic features in some of the series. This is illustrated in Figure 5, which shows a plot of different rules generated from the earthquake sequence discretized with window sizes 2, 4, 8, and 16. This figure shows that by increasing the window size, the generated rules zoom in on the part of the sequence representing an earthquake.

We observed the same effect in the sunspot set, but it was not as apparent in the ECG series (data not shown). This is most likely because the ECG sequence contains far less noise than the earthquake and sunspot sequences. To test this hypothesis, several versions of the ECG series with increasing noise levels were constructed. These sequences were constructed by adding Gaussian noise with mean zero and a the standard deviations set to $0.1\%, 0.5\%, 1\%, 5\%, 10\%$ and 20% of the original value range.

The system was able to generate good rules recognizing the characteristic feature of the ECG series for all window sizes for the two lowest noise levels. For 1% error level, the system only generated good rules for sequences discretized with window sizes of 4 or more. For the other series, the minimum window size required for generating good rules was 8, 16 and 32, respectively (in order of increasing noise levels).

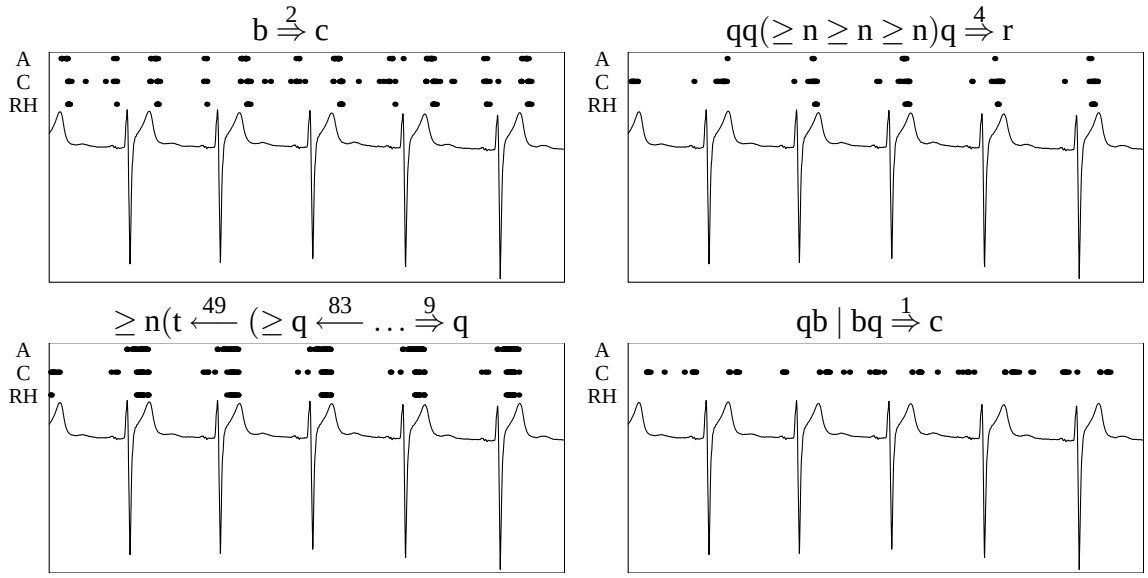

Figure 4: Hit locations in a subsequence of the ECG series of the rules from Table 2 (see Figure 2 for the definitions of *A*, *C*, and *RH*).

When analyzing the network series with increasing window sizes, the same effect was not observed (data not shown). A possible explanation is that the significant feature in the network series is short bursts of increased network activity, represented by isolated series of spikes. Increasing the window size increases the minimum distance between the antecedent and consequent (see section 2.5). In addition, large window sizes represent more long term trends in the data (see section 2.1). Thus one should expect that rules produced from large window sizes will focus on more long term trends than rules produced from small window sizes. As shown above, when the window size is increased, the long term trends in the sunspot, earthquake, and noisy ECG data are more easily detected, while the short bursts in the network series are not. Thus the results support this proposition.

4 Summary and Conclusion

An algorithm for unsupervised data mining in time series has been presented. The algorithm works by optimizing several, often conflicting, measures of rule quality. As a result, it is able to generate a set of rules exploring different aspects of the time series analyzed. This has been demonstrated by analyzing several different sequences, and extracting rules detecting the significant features in each sequence. The robustness to noise has also been demonstrated.

The algorithm presented here is an extension of a method presented in [2], where a single ad hoc rule goodness measure was used. This work follows the more natural approach when dealing with multiple, conflicting objectives, using multiobjective optimization to generate several possible solutions instead of a single result. This leaves the task of evaluating the trade-offs between the different objectives to the human user. In addition, optimizing a single measure may reduce the diversity of the resulting rules, and thereby omit certain potentially interesting results.

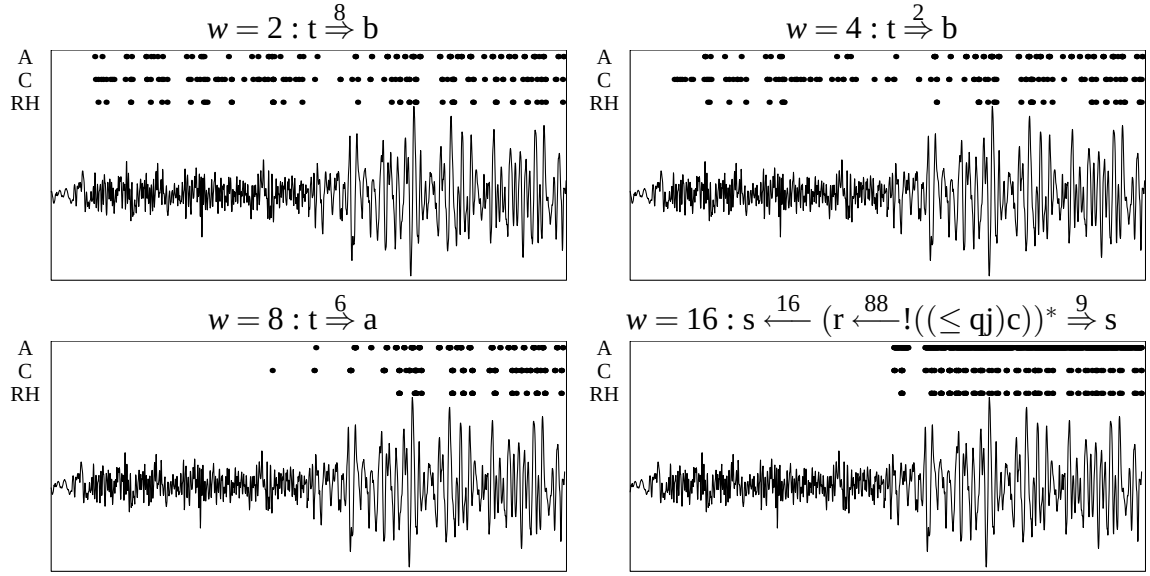

Figure 5: Rules generated from the earthquake series with increasing window sizes (see Figure 2 for the definitions of *A*, *C*, and *RH*).

References

[1] A. A Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Spinger-Verlag, Berlin, 2002.

[2] Pål Sætrom and Magnus Lie Hetland. Mining interesting temporal rules with genetic programming and specialized hardware. In *Proceedings of The 2003 International Conference on Machine Learning and Applications (ICMLA'03) (to appear)*, 2003.

[3] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[4] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[5] Carlos A. Coello Coello. A short tutorial on evolutionary multiobjective optimization. In *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[6] Carlos A. Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2):109–143, 2000.

[7] R. J. Hilderman and H. J. Hamilton. Knowledge discovery and interestingness measures: A survey. Technical Report CS 99–04, Department of Computer Science, University of Regina, Saskatchewan, Canada, October 1999.

[8] P. Smyth and R. M. Goodman. Rule induction using information theory. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 159–176. MIT Press, Cambridge, MA, 1991.

[9] Fast Search & Transfer ASA. Digital processing device.

[10] Fast Search & Transfer ASA. Søkeprosessor.

[11] E. Keogh and T. Folias. The UCR time series data mining archive. `http://www.cs.ucr.edu/~eamonn/TSDMA`, sep 2002.

[12] Interagon AS. The Interagon query language : a reference guide. `http://www.interagon.com/pub/whitepapers/IQL.reference-latest.pdf`, sep 2002.