Combinatorial ECG Analysis for Mobile Devices

Costas S. Iliopoulos[*]
Algorithm Design Group
Department of Computer Science
King's College London
Strand, London WC2R 2LS, England
csi@dcs.kcl.ac.uk

Spiros Michalakopoulos[†]
Algorithm Design Group
Department of Computer Science
King's College London
Strand, London WC2R 2LS, England
spiros@dcs.kcl.ac.uk

ABSTRACT
A combinatorial model for analyzing and interpreting an electrocardiogram (ECG) was presented in [Allali et al. '10] and [Iliopoulos & Michalakopoulos '09 and '10]. An application of the model is $\mathcal{QRS}$ peak detection and the resulting algorithm was shown to be space, as well as time efficient. Experimental results on the MIT-BIH Arrhythmia database were shown. In this paper, the peak detection algorithm is implemented in Java ME. The program is deployed on a mobile phone and simulators, and experimental results are discussed.

Categories and Subject Descriptors
J.3 [**Life and Medical Sciences**]: Medical information systems

General Terms
Algorithms, Performance

Keywords
ECG analysis, MIT-BIH arrhythmia database, QRS detection, Java ME, Mobile phone development

1. INTRODUCTION
An electrocardiogram (ECG) is obtained by placing electrodes on the skin and measuring the direction of electrical current discharged by the heart. The current is plotted into waveforms and displayed as in Figure 1.

A *lead* provides a view of the heart's electrical activity between one positive and one negative pole [16, 18]. Most standard ECG recordings are obtained using a 12-lead device, in clinical settings, and a 2-lead device, in Holter (ambulatory) monitors. The sample in Figure 1 is from a 2-lead reading, as are all 48 records in the MIT-BIH Arrhythmia database [13] that was used for development and testing purposes.

There are many algorithms [14, 7], and software that interpret electrocardiograms. A comparative study [12] identified a number of algorithmic approaches used, such as neural networks [8], digital filters/wavelet transform [1, 11] and syntactic algorithms [19, 15]. Some of these algorithms have been implemented and tested with remarkable results, most notably [1].

The novel approach taken here has similarities to the syntactic methods. Using a simple alphabet the algorithm reads the signals and from the differences between consecutive electrical potentials builds sequences of characters, as shown in Figure 2. The advantage of the approach developed here is the low space (RAM) requirements, as opposed to the memory demands in traditional signal processing algorithms.

Holter monitors normally record ECG data for a period of 24 or 48 hours. Recent advances in mobile device technology have made it possible for cellular phones and other handheld devices to be considered for recording ECG data [17, 4]. The recorded data can be compressed [5, 6], and either held on the device or packaged and sent. In the case of cellular phones, the ECG data could be automatically transmitted at regular times, or when certain exceptional program conditions are met.

2. DEFINITIONS
A *signal* s is a k-tuple $(t, p_{MLII}, p_{V1}, ...)$, where t is the time in seconds and p_E is the electrical potential at lead E in millivolts. When the signal read is from a single lead, it is represented as a pair, a 2-tuple, (t, p).

A *sequence of signals* $s_1, s_2, \ldots, s_n$ is a sequence of pairs, (t_1, p_1), (t_2, p_2), $\ldots$, (t_n, p_n). The readings are taken at regular time *intervals*, the *sample rate* σ. Note that $\sigma = t_{i+1} - t_i = t_i - t_{i-1}$, for all $i \in [2, n-1]$.

Let $\hat{\Sigma}$ and Σ be the following alphabets:

$$\hat{\Sigma} = \{--, -, 0, +, ++\}$$

$$\Sigma = \{C^{--}, C^{-}, C^{0}, C^{+}, C^{++}\}$$

[*]Digital Ecosystems & Business Intelligence Institute, Curtin University, Perth, Australia.

[†]Supported by the Engineering and Physical Sciences Research Council (EPSRC) under the Doctoral Training Award (DTA).

MIR'10, March 29–31, 2010, Philadelphia, Pennsylvania, USA.

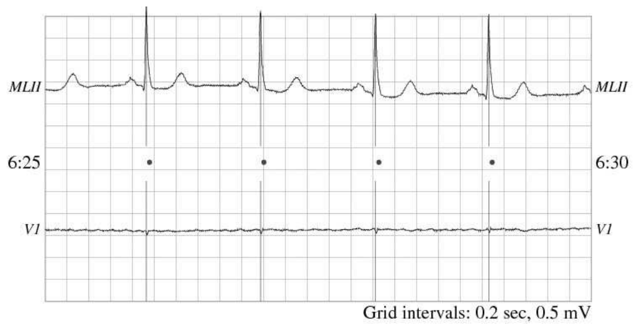

Figure 1: Short sample of ECG for record 101 from the MIT-BIH arrhythmia database.

and $\mathcal{A}$ the set of non-trivial pairs:

$$\mathcal{A} = \{(+,0), (-,0), (++,+), (--,-)\}$$

Note that trivial pairs are $(0,0), (+,+), ...$

The *rate of change* is the difference between the potentials of two consecutive signals. Thus, at position i, the rate of change is $p_i - p_{i-1}$. A gradual rate *increase* is depicted as C^{+}, a sharp rate increase as C^{++} and gradual and sharp *decreases* equivalently as C^{-} and C^{--}. A negligible or zero rate change is C^{0}.

Two parameters, a *low threshold* μ_E and a *high threshold* μ'_E are used to filter the rates of change for readings from lead E. For single lead readings, the thresholds are denoted as μ and μ'. For example, in the MIT-BIH Arrhythmia database (the test db), it has heuristically been determined that reasonable values for many records are $\mu_{MLII} = 0.01$ and $\mu'_{MLII} = 0.05$.

The *direction* of the rate of change is a character d in Σ. To determine d_i, Equation 1 is used:

$$d_i = \begin{cases} C^0, & \text{if } |p_{i+1} - p_i| \le \mu \\ C^-, & \text{if } -\mu' < p_{i+1} - p_i < -\mu \\ C^+, & \text{if } \mu < p_{i+1} - p_i < \mu' \\ C^{--}, & \text{if } p_{i+1} - p_i \le -\mu' \\ C^{++}, & \text{if } p_{i+1} - p_i \ge \mu' \end{cases} \quad (1)$$

Two consecutive rate changes have *direction equivalence*, when there's a smooth transition from one rate change to the other, without a significant change of direction.

A consecutive sequence of direction equivalent rate changes is defined as $\mathcal{C}[x,k]$ of C^x, where $x \in \hat{\Sigma}$, $k \in \mathbb{N}^+$ and for some $C^x \in \Sigma$. A *direction equivalent series* $\mathcal{C}[x,k]$, is referred to simply as a *series* when it is implicit from the context.

Formally, C^x *is equivalent to* C^y,

$$C^x \sim C^y \quad (2)$$

if $(x,y) \in \mathcal{A}$. Furthermore, the two sequences are equivalent

$$C^{x_1}C^{x_2}...C^{x_k} \approx C^{y_1}C^{y_2}...C^{y_k} \quad (3)$$

if $C^{x_i} \sim C^{y_i}$, $\forall i$.

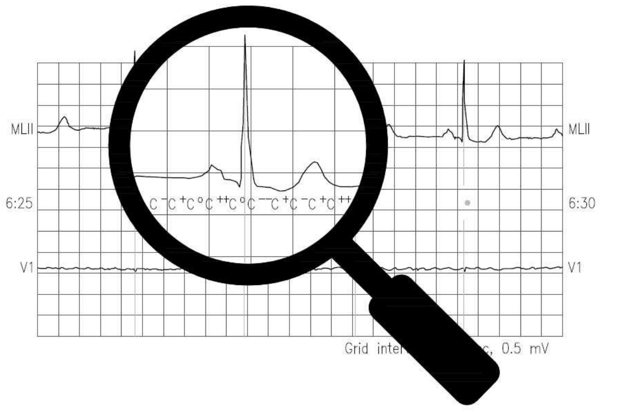

Figure 2: The differences between consecutive ECG signals define letters from alphabet $\Sigma = \{C^0, C^+, C^-, C^{++}, C^{--}\}$.

Also note, $\mathcal{C}[+,k] \approx C^{x_1}C^{x_2}...C^{x_k}$, where $x_i \in \{+,0\}$, for $i \in [1..k]$, $\mathcal{C}[++,k] \approx C^{x_1}C^{x_2}...C^{x_k}$ where $x_i \in \{++,+\}$ for $i \in [1..k]$ and similarly for $\mathcal{C}[-,k]$ and $\mathcal{C}[--,k]$.

Finally, the formal definition of the $\mathcal{QRS}$ Detection Problem is:

PROBLEM 1 ($\mathcal{QRS}$ DETECTION). *Given an electrocardiogram as a sequence of signals $s_1, s_2 \ldots, s_n = (t_1,p_1), (t_2,p_2), \ldots, (t_n, p_n)$, where t_i, $p_i \in \mathbb{R}$, detect the $\mathcal{QRS}$ complexes and the $\mathcal{RR}$ intervals.*

3. IMPLEMENTATION

The implementation for use on mobile devices was done in Java ME (Java Platform, Micro Edition). This was chosen because of the extensive availability of the Java platform on mobile phones and other hand-held devices. From a programming perspective, Java ME is simply a cut-down version of Java, with a reduced set of API's.

The initial implementation, in [9, 10] was done in C++. The program was tested on all 48 records in the test db, and the experimental results were concerned with the accuracy of peak detection. The results were promising, in the range of 80% overall, with higher than 95% on "normal" records. As discussed in the papers, given improved learning period processing, better accuracy results should be achievable on the more "difficult" records.

For the mobile device version of the program, the C++ code was initially ported to Java. Although there were very few challenges, given the similarities in the programming languages, the differences in the running speed are significant, due to the fact that C++ is a compiled language, whereas Java is partly interpreted. Table 1 shows a comparison of the runtime on 3 records from the test db.

The Java program was tested for correctness against the C++ version, by comparing the percentages of accuracy of $\mathcal{QRS}$ peak detection. The translation was found to be accurate, because the percentages were identical. This was the

Table 1: Running time of the program on records 100, 106 and 119 in the test database. Each record has length 30 mins.

Record	C++	Java
100	5.2 sec	6.88 mins
106	5.1 sec	6.84 mins
119	5.6 sec	7.61 mins

most important aspect of the porting, since the ultimate goal was not a Java version of the program, but a Java ME version. Thus, the Java code was not optimized, which may account partially for the large difference in runtime speed from the C++ version.

The program was then ported to Java ME. In the original implementation, which was not overly concerned with storage details, the ECG signal files were read in and processed, using a vector $v_{\mathcal{R}}$, which was not reallocated until the program finished processing a record. For the mobile device version, only the necessary data need be read in and processed for each window of length ω , and then discarded from the heap. Thus, a modification to the algorithm was to use a queue data structure $q_{\mathcal{R}}$ instead of a vector.

Another implementation detail was the simulation of real world conditions. The program was designed with the vision of the data being input to the phone, via USB connection, or possibly Bluetooth technology. This requires hardware in the form of ECG sensors that transmit this data and communicate with the phone. Because of the current difficulties in obtaining such devices, an attempt to simulate real-world conditions was made, by delaying the reading of each signal by the sample rate σ in the cases where the reading by the program occurs faster than σ. This is generally the case in the emulators that the program was tested on, but not in the case of the mobile phone the program was run on. For more information about the tests, please see Section 5.

4. THE ALGORITHM

An outline of the algorithm:

STEP 1: During the learning period determine values μ, μ', *rr_max*, *rr_min*, *pot_min*, *pot_max*, k_1 and k_2 for the subject.

STEP 2: Determine d_i by difference in potential from last signal, as in Equation 1.

STEP 3: Either add d_i to current series $\mathcal{C}[x, k]$ or start a new series if certain conditions met. If added to current series and identifies a normal beat, then store in $v_{\mathcal{R}}$.

STEP 4: Repeat steps 2 and 3 until end of electrocardiogram.

In step 3 it is stated that a new series is started if certain conditions are met. The next paragraphs describe what these conditions are and how they are met.

The algorithm holds 5 counters, one for each character in Σ, i.e. $+_{count}$, $-_{count}$, 0_{count}, $--_{count}$ and $++_{count}$. The appropriate counter is incremented when the current series is $\mathcal{C}[x, k]$ and d_i is not equivalent to x, i.e. if $d_i = C^y \nsim C^x$, the y counter (y_{count}) is incremented. All the counters are zeroed when a new series is initiated.

To start a new series, d_i's counter must be above a certain number ν. This means that the current series $\mathcal{C}[x, k]$ is only terminated after ν characters $y_1, y_2, ..., y_\nu$, such that $y_1 = y_2 = ... = y_\nu$ and $y_i \nsim x$ are read. This implies that whichever non-equivalent character y's counter, y_{count}, reaches ν first, becomes the new x.

Finally, again in step 3, if $d_i \sim x$ a check must be made whether a normal beat has been identified. In effect, what is checked is whether the time of this beat has occurred within *rr_min* and *rr_max* of the last identified beat. Pseudo-code for the algorithm is depicted in Algorithm 1.

From $q_{\mathcal{R}}$, the time difference between two $\mathcal{R}$ waves, the $\mathcal{RR}$ *interval* is calculated, $q_{\mathcal{R}_i} - q_{\mathcal{R}_{i-1}}$. The heart rate is calculated by averaging a number, for example 3 in the test db, of successive $\mathcal{RR}$ intervals.

Algorithm 1 "Pure" Online Process ECG

```
 1: function PUREONLINEPROCESSECG
 2:     $\mathcal{C}[x,k] \leftarrow [0,0]$                ▷ current series
 3:     $\mathcal{C}'[x,k] \leftarrow [0,0]$               ▷ previous series
 4:     while not end of ECG do
 5:         calculate $d_i$ as in Equation 1
 6:         if $d_i \sim C^x$ then
 7:             increment current series
 8:             if normal beat then
 9:                 push beat onto $q_{\mathcal{R}}$
10:                 calculate heart rate
11:                 pop last element off $q_{\mathcal{R}}$
12:         else
13:             increment count: $y_{count} \leftarrow y_{count} + 1$
14:             if $y_{count} \geq \nu$ then              ▷ new series
15:                 $\mathcal{C}'[x,k] \leftarrow \mathcal{C}[x,k]$
16:                 $\mathcal{C}[x,k] \leftarrow [y, y_{count}]$
17:                 zeroise all counters
```

5. EXPERIMENTAL RESULTS

The algorithm was implemented, and tests were carried out on records from the test db. The implementation was tested on two virtual devices, the Sun Java(TM) Wireless Toolkit 2.5.2 for CLDC emulator, and Nokia Series 40 6th Edition SDK emulator, and one physical device, the Nokia 2630 mobile phone.

The tests were performed on various records and the accuracy of the translation from C++ to Java to Java ME was confirmed. Thus, in evaluating the performance of the program the only concern is the speed of execution. Table 2 shows a comparison between the running times of the three devices.

As discussed previously, in the "real world", the ECG data would be read into the device via some form of ECG sensors. It would be processed in chunks and discarded, because of low memory capacity in hand-held devices. The tests were performed with data from the test db, by embedding 1 min chunks of the file into the program.

Table 2: Experimental results on 1 min segments of record 100 from test db. The sample rate $\sigma = 1000/360$ ms.

Device	Speed (sec)	Compared to σ
Sun emulator	8.651	7 times faster
Nokia emulator	41.286	1.5 times faster
Nokia phone	235.914	4 times slower

5.1 The Demos

For the purposes of the demos[1], the execution speed has been slowed down for the emulators, by attempting to simulate the reading in of data at the same sample rate as in the test database, $\sigma = 1000/360$ ms. They show the execution of roughly 30 sec of data from record 100.

A heart rate is considered "normal" if it lies between 50 and 110 beats per minute [3]. In Algorithm 1, a positive beat detection is considered *valid* if it results in a 50 to 110 bpm heart rate i.e., if it is between 0.55 and 1.2 sec from the previous beat detection. The heart rate in the implementation is calculated by averaging each 3 consecutive valid beats.

6. CONCLUSIONS & FURTHER WORKS

In this paper, the $\mathcal{QRS}$ peak detection algorithm of [9, 10] was modified and implemented for mobile devices. Java ME was chosen because of the wide availability of the Java platform on devices. Tests were carried out and experimental results were discussed, which show that the current technological limitations, possibly combined with the chosen language and implementation, fail to deliver a useful real-world solution.

It is believed however, that the model remains promising. The Java code could be optimized, which would significantly decrease the running time. However, it is felt that the factor is too large, 4 times slower, than can be achieved by code optimization alone.

A solution which could possibly achieve the objective would be to implement the algorithm in a different language and for different platforms. Java is inherently slow and a comparison of the same program written in C++ as opposed to Java, albeit not for hand-held devices, suggests that the execution speed for the specific program is faster by a magnitude of 80 times for the compiled C++ version.

[1] at the time of writing, these can be seen at: http://www.dcs.kcl.ac.uk/pg/spiros/videos/sun-demo.htm, and http://www.dcs.kcl.ac.uk/pg/spiros/videos/nokia-demo.htm

7. REFERENCES

[1] V. X. Afonso, W. J. Tompkins, T. Q. Nguyen, and S. Luo. Ecg beat detection using filter banks. *Biomedical Engineering, IEEE Transactions on*, 46(2):192–202, Feb. 1999.

[2] J. Allali, P. Ferraro, C. S. Iliopoulos, and S. Michalakopoulos. Combinatorial detection of arrhythmia. In *Proceedings of the 3rd International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2010)*, 2010. (to appear).

[3] ANSI/AAMI. *EC38: Ambulatory Electrocardiographs.* Association for the Advancement of Medical Instrumentation, 1998.

[4] X. Chen, C. T. Ho, E. T. Lim, and T. Z. Kyaw. Cellular phone based online ecg processing for ambulatory and continuous detection. *Computers in Cardiology*, pages 653–656, Sept. 2007.

[5] J. A. Crowe, N. M. Gibson, M. S. Woolfson, and M. G. Somekh. Wavelet transform as a potential tool for ecg analysis and compression. *Biomed. Eng.*, 14(3), 1992.

[6] M. L. Hilton. Wavelet and wavelet packet compression of electrocardiograms. *IEEE Trans. Biomed. Eng*, 44:394–402, 1997.

[7] Y. H. Hu, S. Palreddy, and W. J. Tompkins. A patient-adaptable ecg beat classifier using a mixture of experts approach. *Biomedical Engineering, IEEE Transactions on*, 44(9):891–900, 1997.

[8] Y. H. Hu, W. J. Tompkins, J. L. Urrusti, and V. X. Afonso. Applications of artificial neural networks for ecg signal detection and classification. *Journal of Electrocardiology*, 26, 1994.

[9] C. S. Iliopoulos and S. Michalakopoulos. A combinatorial model for ecg interpretation. In *Proceedings of ICMIBE 2009, International Conference on Medical Informatics and Biomedical Engineering.* World Academy of Science, June 2009.

[10] C. S. Iliopoulos and S. Michalakopoulos. A combinatorial model for ecg interpretation. *International Journal of Biological and Life Sciences*, 02(1), 2010.

[11] H. Inoue and A. Miyazaki. A noise reduction method for ecg signals using the dyadic wavelet transform (special section of papers selected from itc-cscc'97). *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 81(6):1001–1007, 1998.

[12] B. U. Kohler, C. Hennig, and R. Orglmeister. The principles of software qrs detection. *Engineering in Medicine and Biology Magazine*, 21(1):42–57, 2002.

[13] Massachusets Institute of Technology. Mit-bih ecg database. 1999. Available: http://ecg.mit.edu/.

[14] G. Moody and R. Mark. Development and evaluation of a 2-lead ecg analysis program. *Computers in Cardiology*, pages 39–44, 1982.

[15] G. Papakonstantinou, E. Skordalakis, and F. Gritzali. An attribute grammar for qrs detection. *Pattern Recogn.*, 19(4):297–303, 1986.

[16] Springhouse (Editor). *ECG Interpretation Made Incredibly Easy!* Lippincott Williams & Wilkins, 4th edition, 2007.

[17] F. Sufi, Q. Fang, and I. Cosic. Ecg r-r peak detection on mobile phones. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 3697–3700, 2007.

[18] M. S. Thaler. *The Only EKG Book You'll Ever Need.* Lippincott Williams & Wilkins, 5th edition, October 2006.

[19] J. K. Udupa and I. S. N. Murthy. Syntactic approach to ecg rhythm analysis. *IEEE Trans. Biomed. Eng.*, BME-27(7):370– 375, July 1980.