

# Modelling Medical Time Series Using Grammar-Guided Genetic Programming

Fernando Alonso, Loïc Martínez, Aurora Pérez, Agustín Santamaría,  
and Juan Pedro Valente

Facultad de Informática. Universidad Politécnica de Madrid. Campus de Montegancedo.  
28660 Boadilla del Monte. Madrid. Spain

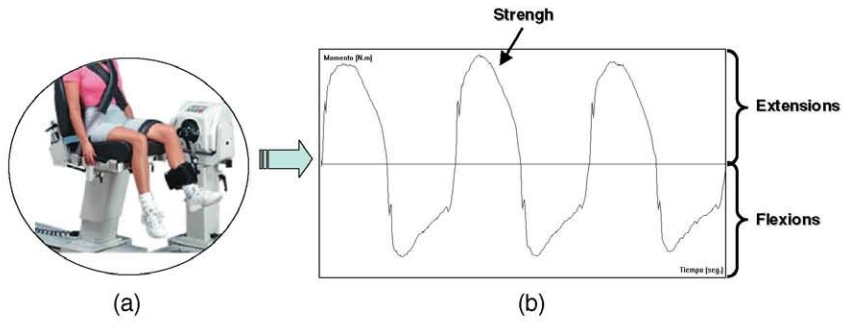
**Abstract.** The analysis of time series is extremely important in the field of medicine, because this is the format of many medical data types. Most of the approaches that address this problem are based on numerical algorithms that calculate distances, clusters, reference models, etc. However, a symbolic rather than numerical analysis is sometimes needed to search for the characteristics of time series. Symbolic information helps users to efficiently analyse and compare time series in the same or in a similar way as a domain expert would. This paper describes the definition of the symbolic domain, the process of converting numerical into symbolic time series and a distance for comparing symbolic temporal sequences. Then, the paper focuses on a method to create the symbolic reference model for a certain population using grammar-guided genetic programming. The work is applied to the isokinetics domain within an application called I4.

**Keywords:** Time series characterization, isokinetics, symbolic distance, information extraction, reference model, text mining.

## 1 Introduction

An important domain for the application of time series analysis in the medical field is physiotherapy and, more specifically, muscle function assessment based on isokinetics data.

Isokinetics data is retrieved by an isokinetics machine (Fig. 1a), on which patients perform exercises using any of their joints (knee, elbow, ankle, etc.) at maximum strength. To assure that the patient exercises at constant speed, the machine puts up the required resistance to the strength the patient exerts. As our patients are chiefly sportspeople, we decided to focus on knee exercises (extensions and flexions) since most of the data and knowledge gathered by sports physicians is related to this joint. The data takes the form of a strength curve with additional information on the angle of the knee (Fig. 1b). The positive values of the curve represent extensions (knee angle from  $90^\circ$  to  $0^\circ$ ) and the negative values represent flexions (angle from  $0^\circ$  to  $90^\circ$ ).



**Fig. 1.** Isokinetics machine (a) and collected data (b)

After observing experts at work, we found that they apply their knowledge and expertise to focus on certain sections of the isokinetics curve and ignore others. Therefore, we looked for a way of bringing this machine's output closer to the information sports physicians deal with in their routine work, since they demand a representation related to their own way of thinking and operating. Hence, symbolic series have been used as an option that more closely resembles an expert's conceptual mechanisms.

To do this, our research focused primarily on the design of the symbols extraction method that translates numerical time series into symbolic temporal series. Then, we designed an isokinetics symbolic distance measure to indicate how similar two symbolic time series are. This way, symbolic sequences can be automatically compared to detect similarities, class patients, etc. Finally, we applied data mining (DM) techniques based on grammar-guided genetic programming (GGGP) to create reference models useful for defining population groups.

Section 2 of the paper gives an overview of the I4 (Intelligent Interpretation of Isokinetics Information) system, of which this research is part. Section 3 describes the symbol extraction method. Section 4 explains the symbolic distance measure. Section 5 describes the use of genetic algorithms to create symbolic models from symbolic time sequences. Finally, section 6 presents the research results and section 7 outlines some conclusions, and mentions future lines of research.

## 2 I4 System Overview

The I4 System provides sports physicians with a set of tools to analyze patient strength data output by an isokinetics machine. It is composed of a data preparation subsystem, a Knowledge-Based System (KBS), a numerical Knowledge Discovery in Databases (nKDD) subsystem, a symbolic Knowledge Discovery in Databases (sKDD) subsystem (objective of this paper), and a Visualization module (Fig. 2). The data preparation subsystem manages the tasks of translating, formatting, cleaning and pre-processing the time series obtained from the isokinetics data. These tasks use expert knowledge and generate a database in which data is homogeneous, consistent and noise free. The KBS module analyses this data to make it easier for novice users and also blind physiotherapists to interpret the isokinetics curves. The nKDD performs

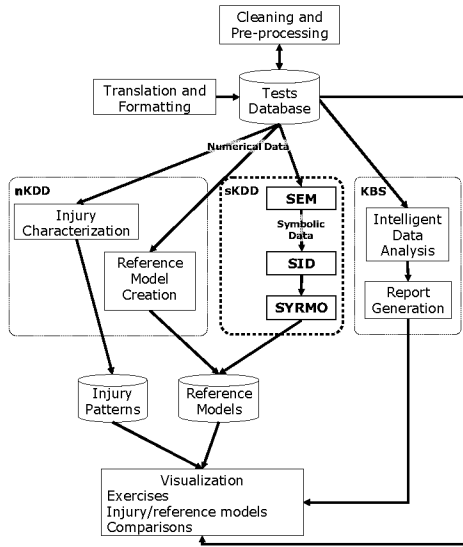


Fig. 2. I4 System Overview

data mining on the numerical isokinetics data to define reference models for patient groups and to identify injury patterns. Finally, the Visualization module displays exercises, injury patterns, reference models, etc.

Many of these functionalities are used on a daily basis by specialized physicians to assess the potential of their patients (mostly top-competition sportsmen and women), diagnose injuries and analyse what progress patients have made in injury recovery. The system is reliable and outputs equivalent results to what an expert would. However, it has failed to gain experts' total confidence. This is because the information the expert receives from the I4 system does not highlight the significant aspects of the isokinetics series in a language that they can easily understand. This has led to the need to build a symbolic Knowledge Discovery in Databases subsystem (sKDD) to solve this problem. The sKDD subsystem contains: a Symbolic Extraction Method (SEM) to extract the symbolic sequence from a numerical series; a Symbolic Isokinetics Distance (SID) module to get a similarity measure between two symbolic isokinetics sequences; and a SYmbolic Reference MOdels (SYRMO) method to create a reference model from a set of isokinetics exercises. The sKDD subsystem should produce results that are equally reliable as the nKDD subsystem, and it should also give a reasonable explanation of the results in terms of the domain under study. This paper focuses on the design of the sKDD.

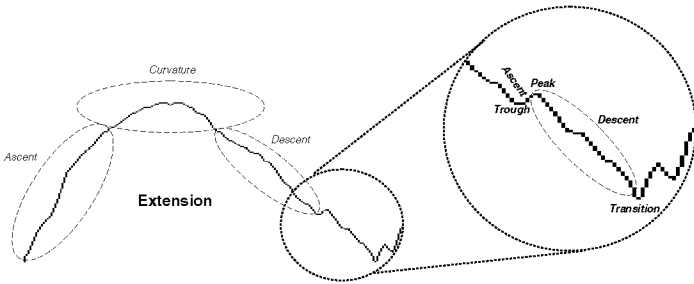
### 3 Conversion of Numerical into Symbolic Time Sequences

To be able to develop a symbolic comparison method it is necessary to translate the numerical time sequences output by the isokinetics machine into symbolic time sequences. The first option was to use SAX (Symbolic Aggregate approXimation)

to do this translation. This method is able to reduce the temporal sequence dimensionality and assures that the symbolic distance used is less than or equal to distance between the original two sequences. However, the symbols output automatically using the Piecewise Aggregate Approximation (PAA) and the Gaussian distribution do not have expert semantic content, that is, they would not be equivalent to the symbols that the expert identifies when analysing the sequence. To do this it is necessary [1] to find out what symbol alphabet the isokinetics expert uses to analyse the temporal sequences.

After the first few interviews, the expert stated that there were two visually distinguishable regions in every exercise: knee extension and flexion. Both had a similar morphology (the shape shown in Fig. 3), from which we were able to identify the following symbols:

- *Ascent*: part where the patient gradually increases the strength applied.
- *Descent*: part where the patient gradually decreases the strength applied.
- *Peak*: a prominent part in any part of the sequence.
- *Trough*: a depression in any part of the sequence.
- *Curvature*: the upper section of a region.
- *Transition*: the changeover from extension to flexion (or vice versa).



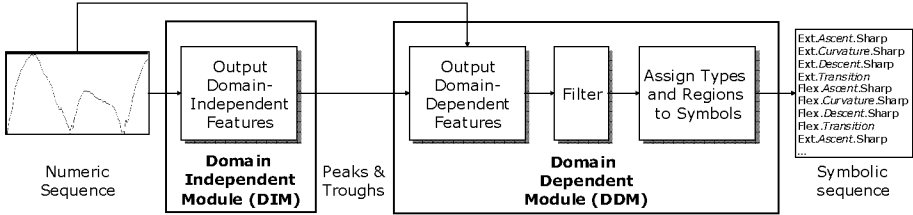
**Fig. 3.** Symbols of an isokinetics curve

Each isokinetics symbol can have different shapes, that is, different types that are taken into account when translating a numerical temporal sequence into a symbolic series. The types were also elicited from the expert as he analyzed test cases that constituted a significant sample of the whole database. As the expert separated an extension from a flexion, each symbol had to be labelled with its type and also with the keyword “Ext” or “Flex”. The set of symbols, types and regions form an alphabet called ISA (Isokinetics Symbols Alphabet), shown in Table 1.

This ISA is used to get symbolic sequences from numerical temporal sequences. The Symbolic Extraction Method (SEM), shown in Fig. 4, was designed to make this transformation. First, a pre-prepared numerical sequence is put through the domain-independent module (DIM), which outputs a set of domain independent features, that is, peaks and troughs. Both the features output by the DIM (peaks and troughs) and the actual numerical sequence data will be used as input for the domain-dependent

**Table 1.** Isokinetics Symbols Alphabet

Region	Symbol	Types		
EXT	<i>Ascent</i>	Sharp	Gentle	
	<i>Descent</i>	Sharp	Gentle	
	<i>Trough</i>	Big	Small	
	<i>Peak</i>	Big	Small	
FLEX	<i>Curvature</i>	Sharp	Flat	Irregular
	<i>Transition</i>	-		



**Fig. 4.** Architecture of SEM

module (DDM). The DDM outputs all the domain-dependent data of the sequence. This module is divided into three submodules:

- *Output of domain-dependent features.* The aim is to get all the symbols that characterize the given numerical sequence. This module selects the relevant peaks and troughs and identifies the ascents, descents and curvatures.
- *Filter.* The set of symbols output by the above submodule is put through a filtering stage. Apart from other filtering processes, this filter checks that there are no consecutive symbols that are equal. For example, it makes no sense to have two ascents one after the other, because they would really be just one ascent.
- *Assign types to symbols.* The goal of this submodule is to label each symbol with a type. This process is based on a set of rules that use a number of thresholds to define the symbol type in each case.

A graphical interface has been designed to easily work with the SEM (see Fig. 5). An exercise is selected as input to the SEM. The original temporal sequence of the exercise is displayed at the top of the interface. The central part displays the translation of the temporal sequence into symbols, illustrating all the SEM stages. The first stage outputs the domain-independent features, as is shown on the left under the heading “FEATURES”. This list contains all the information related to each peak/trough and is formatted as follows:

```
<feature>. Grad:<gradient_value> Start:<initial_value>
End:<final_value> Ampl:<amplitude_value>
Dur:<duration_value> <value_of_the_point>
```

The next stage of the method is to output the domain dependent symbols, which are shown under the heading “DOMAIN-DEPENDENT SYMBOLS”. The threshold

parameters that are used to output these symbols are listed under “FILTERING PARAMETERS”.

The result of the last stage of the SEM is set out on the right side of the interface, under the heading “DOMAIN-DEPENDENT TYPED SYMBOLS”. It is the type characterisation of each symbol. The threshold parameters used are shown as “TYPOLOGY PARAMETERS”.

The curve reconstructed from the symbols is shown at the bottom.

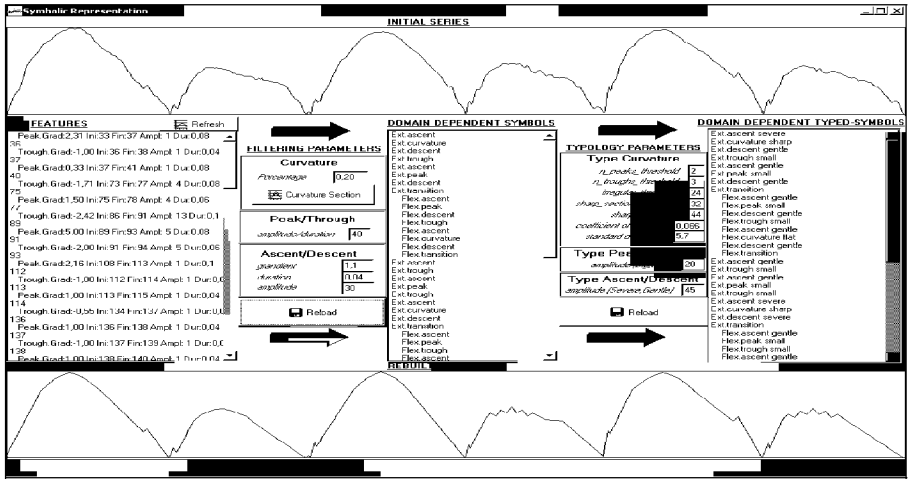


Fig. 5. Symbolic representation interface

As stated by the expert, SEM is an important aid for physicians in writing reports, examining the evolution of an athlete’s joint, diagnosing injuries or controlling treatment after a medical diagnosis.

#### 4 Comparison of Isokinetics Symbolic Sequences

Our next goal is to find a similarity measure that can be used to compare symbolic isokinetics sequences and perform data mining tasks.

After a thorough study to select the best similarity measure for the medical field of isokinetics, we reached the conclusion that a new measure needed to be designed. This measure is based on edit distances, which are the best fit for the isokinetics domain, as they take into account the order of the components and the morphology of the sequence. However, none of the distances we examined exactly fits our problem, because the symbols used in the isokinetics domain also have an associated type that needs to be taken into account to calculate the distances.

This led us to propose a variation on the Needleman-Wunch distance. The suggested distance, the Symbolic Isokinetics Distance (SID), allocates a variable cost to the *insert* and *delete* operations depending on the symbol and symbol type to be

inserted or deleted. It also allocates a variable cost to the *substitute* operations depending on the symbol and type that are substituted.

Fig. 6 shows the three steps required to calculate the SID of two symbolic sequences: calculate the distance between each pair of subsequences, normalize these distances and calculate the arithmetic mean to get the total distance.

The researched isokinetics sequences are composed of three repetitions, and each repetition is composed of an extension and a flexion. Therefore, an isokinetics sequence contains six parts, each of which is represented by the notation shown in (1).

$$\langle \text{Zone} \rangle \langle \text{Repetition} \rangle \langle \text{Sequence} \rangle \quad (1)$$

where  $\langle \text{Zone} \rangle$  can take the value E (for Extension) or F (for Flexion),  $\langle \text{Repetition} \rangle$  can take the value  $R^1$ ,  $R^2$  or  $R^3$  depending on whether it is repetition 1, 2 or 3, and  $\langle \text{Sequence} \rangle$  can take the value  $S^1$  or  $S^2$  depending on the sequence 1 or 2.

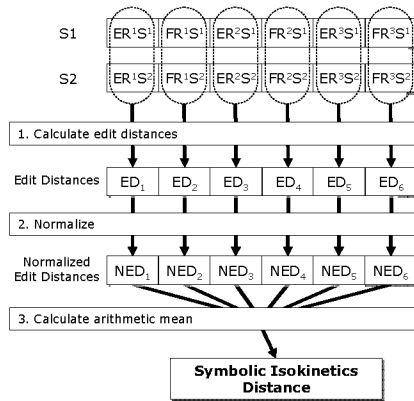


Fig. 6. Computing the Symbolic Isokinetics Distance

The SID between two series, S1, of length n, and S2, of length m, is calculated by building a matrix of  $m \times n$  elements. This matrix includes the accumulated costs of the *insert*, *delete* or *substitute* operations, always calculating the best alignment between the two symbolic sequences for comparison. This prevents trapping in local minima. The value of each matrix element is calculated using equation (2): element (i,j) indicates the SID between S1' and S2' (the subsequences —prefixes— of S1 and S2 ending in elements j and i, respectively); element (m,n) indicates the final SID between S1 and S2. This way, the SID can be used to get the least costly edit command sequence (delete, insert and substitute) for transforming S<sub>1</sub> into S<sub>2</sub>.

$$D(i, j) = \min \begin{cases} D(i-1, j-1) & \text{if } s_i = t_j & // \text{copy} \\ D(i-1, j-1) + \text{SubstituteGapCost} & \text{if } s_i \neq t_j & // \text{substitute} \\ D(i-1, j) + \text{InsertGapCost} & & // \text{insert} \\ D(i, j-1) + \text{DeleteGapCost} & & // \text{delete} \end{cases} \quad (2)$$

Due partly to qualitative aspects (each symbol has a different structural weight) and partly to quantitative issues, not all the operations or all the symbols can be

allocated an identical gapcost in the isokinetics field. For example, curvatures are symbols that are part of any repetition, whereas peaks and troughs are circumstantial symbols, usually induced by minor patient injuries and, therefore, may or may not appear. Additionally, the presence of a large peak cannot be considered the same as there being a small peak. Therefore, each symbol has to be allocated a different weight, and a distinction has to be made depending on the symbol type.

We had to define both the cost of substituting one symbol type by another and the cost of inserting or deleting a particular symbol type. This was done with the help of an isokinetics expert. The *insert* and *delete* costs were unified to assure that the comparison of two series is symmetric.

For the *substitute* cost, several possibilities were weighed up. Initially, we designed a tabular structure, where the table rows and columns included all the symbol types, and the cell (i,j) represented the cost of substituting the symbol type i by the symbol type j. However, this table was hard work for the expert to build. For instance,  $(n \times m)^2 / 2 - (n \times m)$  values are needed if the number of symbols is n and the mean number of types per symbol is m (the table is symmetric and the cost is 0 along the main diagonal). Additionally, this table is not very open to the entry of any change in the symbols alphabet, as the expert would have to put in a lot of work to reformulate the table to accommodate the changes.

To overcome these two problems, we opted for a graph structure, where the principal cost of substituting two symbols is determined mainly by the symbol, whereas the symbol type serves to further specify that cost. Fig. 7b shows this substitution graph. The expert will have to define  $n^2 / 2 - n + n \times m$  values, which is clearly fewer than for the table. Additionally, this structure is much more open to the entry of any changes in the symbols alphabet, and it is also more self-explanatory for the expert.

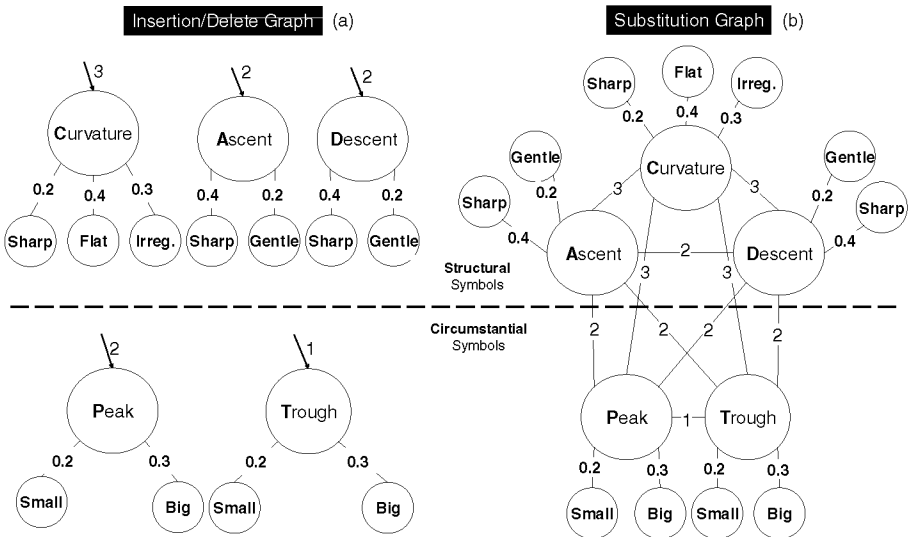


Fig. 7. Insertion/Deletion and Substitution Graph



For the sake of coherency, we have used a similar representation for the *insert* and *delete* costs (Fig. 7a), although, in this case, there is no difference in the number of values that the expert has to define for the graph and for the table.

To make things easier for the expert, we took the graphical representation for each symbol type and defined some initial costs by comparing the area each symbol covered. These initial values were presented to the expert as a starting point.

The gapcosts used in (2) are plotted in the graphs shown in Fig. 7. It is clear from these graphs that there is a cost per symbol to which a cost per type associated with each symbol is added.

The normalization process is applied to the distances between each of the six components of the two sequences for comparison (these distances are denoted  $ED_x$  in Fig. 6, where  $x$  is the number of the component that has been compared). Then all the distances are defined in the interval  $[0, 1]$ . The normalization is based on dividing the obtained distance value by what would have been output in the worst case. In our domain, as all the sequences have six curvatures (two for each repetition), the worst case would be to have *substitute* operations for the curvatures and *substitute* operations for ascents or descent with the worst gapcost.

Once the normalized distances have been obtained for each component, their arithmetic mean is calculated. This outputs the symbolic isokinetics distance between the two compared sequences. This is useful for comparing symbolic time series with reference models to detect injuries or class a sportsperson in a give population group.

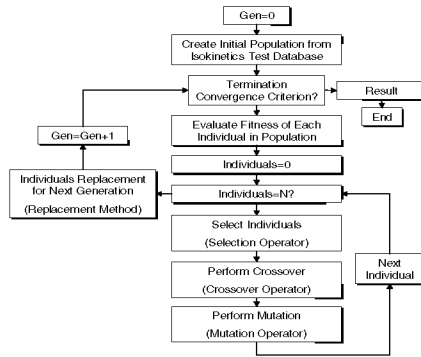
## 5 Using GGGP to Create Reference Models

As part of the research, a method, called SYRMO (SYmbolic Reference MOdels), has been defined to create symbolic reference models from symbolic temporal sequences. A symbolic reference model will be composed of a repetition, that is, an extension and flexion. To do this, we weighed up several alternatives.

The first possibility was to formulate a brute-force algorithm that generated all the possible symbol combinations to form a repetition that would be used as a trial model. Each trial model would be compared with what is currently the best model using the mean of all the symbolic distances of the model with all the existing repetitions. If the mean of the trial model is less than the mean of the existing model, then the model is updated. Evidently, this alternative is not feasible because, apart from producing a combinatorial explosion of symbols, there is no way of generating optimal trial solutions to form a trial model. Trial solutions are generated completely at random, and better and worse trial models are likely to appear in each algorithm run.

Then we weighed up the possibility of using the brute-force algorithm with an improved trial model generation mechanism. This mechanism used the SID algorithm to find the absolute alignments between the symbolic sequences from which the model was to be generated. Our research confirmed that this algorithm was hard to generalize for  $n$  symbolic sequences and did not assure the correct generation of trial models.

Finally, GGGP was applied to create reference models. GGGP is an extension of genetic programming. Genetic programming (GP) is a means of automatically generating computer programs by employing operations inspired by biological evolution [6]. First, the initial population is generated, and then genetic operators, such as



**Fig. 8.** Genetic Programming Algorithm

selection, crossover, mutation and replacement, are executed to breed a population of trial solutions that improves over time

In this case, the initial population is composed of a selected set of symbolic sequences on which the set of genetic operators should be applied. Fig. 8 shows the genetic algorithm's steps. As the process advances, the individuals, in this case symbolic temporal sequences, are selected, crossed and mutated generation by generation to improve a fitness function, applying a generational replacement method. The algorithm ends when the convergence criteria are met. The result is an individual, the best individual in the last population, which is the model of the initial population.

In GP all algorithms start with the random generation of the initial population, which is composed of individuals that represent possible solutions to the search problem at hand. The main disadvantage of this process is that, being completely random, it can cause the generation of invalid individuals: sequences that are too large or that do not represent possible solutions to the problem. One way to overcome this drawback is to keep on generating individuals, discarding the invalid ones, until the initial population is complete. However, the computational cost of this approach is extremely high for problems requiring large population sizes. The goal of grammar-guided genetic programming is to solve the closure problem. This problem involves always generating valid individuals (points or possible solutions that belong to the search space), which directly affects the initial population-generating algorithm. To solve the closure problem, GGGP employs a context-free grammar (CFG) that establishes a formal definition of the syntactical restrictions of the problem to be solved and its possible solutions. Each of the individuals handled by GGGP is a derivation tree that generates and represents a sentence (solution) belonging to the language defined by the CFG

In this case, the isokinetics language is defined by the IG (Isokinetics Grammar) CFG that uses the ISA alphabet, shown in (3).

Once the IG grammar had been defined, we went on to determine the fitness function according to which the individuals of the population were to evolve. After running several experiments, the chosen fitness function was the mean of an individual's symbolic distances from 80% of its nearest neighbours in the initial population. This individual is the reference model that is representative of the initial population.

Then we went on to select the best set of genetic operators and convergence criteria for the GGGP algorithm to converge as fast as possible. To do this, we had to run experiments on different genetic operators with different convergence criteria to find out how the algorithm converged and what percentage of fitness the fitness function achieved.

$$IG = (\sum_N, \sum_T, \text{Exercise}, P)$$

$\sum_N = \{\text{Exercise, Ext, Flex, Climb, Fall, preAsc, preDesc, prePeak, preTrough, postAsc, postDesc, postPeak, postTrough, Curvature, Asc, Desc, Peak, Trough}\}$

$\sum_T = \{\text{Curv-sharp, Curv-flat, Curv-irregular, Asc-sharp, Asc-gentle, Desc-sharp, Desc-gentle, Peak-big, Peak-small, Trough-big, Trough-small, Transition}\}$

$$\begin{aligned}
 P = \{ & \text{Exercise} \rightarrow \text{Ext Transition Flex} \\
 & \text{Ext} \rightarrow \text{Climb Curvature Fall} \\
 & \text{Flex} \rightarrow \text{Climb Curvature Fall} \\
 & \text{Climb} \rightarrow \text{preAsc Asc postAsc} \mid \text{preAsc Asc} \mid \text{Asc postAsc} \mid \text{Asc} \\
 & \text{Fall} \rightarrow \text{preDesc Desc postDesc} \mid \text{preDesc Desc} \mid \text{Desc postDesc} \mid \text{Desc} \\
 & \text{preAsc} \rightarrow \text{preDesc Desc} \mid \text{prePeak Peak} \mid \text{preTrough Trough} \mid \text{Desc} \\
 & \quad \mid \text{Peak} \mid \text{Trough} \\
 & \text{preDesc} \rightarrow \text{preAsc Asc} \mid \text{prePeak Peak} \mid \text{preTrough Trough} \mid \text{Asc} \mid \text{Peak} \\
 & \quad \mid \text{Trough} \\
 & \text{prePeak} \rightarrow \text{preAsc Asc} \mid \text{preDesc Desc} \mid \text{preTrough Trough} \mid \text{Asc} \mid \text{Desc} \\
 & \quad \mid \text{Trough} \\
 & \text{preTrough} \rightarrow \text{preAsc Asc} \mid \text{preDesc Desc} \mid \text{prePeak Peak} \mid \text{Asc} \mid \text{Desc} \\
 & \quad \mid \text{Peak} \\
 & \text{postAsc} \rightarrow \text{Desc postDesc} \mid \text{Peak postPeak} \mid \text{Trough postTrough} \mid \text{Desc} \\
 & \quad \mid \text{Peak} \mid \text{Trough} \\
 & \text{postDesc} \rightarrow \text{Asc postAsc} \mid \text{Peak postPeak} \mid \text{Trough postTrough} \mid \text{Asc} \\
 & \quad \mid \text{Peak} \mid \text{Trough} \\
 & \text{postPeak} \rightarrow \text{Asc postAsc} \mid \text{Desc postDesc} \mid \text{Trough postTrough} \mid \text{Asc} \\
 & \quad \mid \text{Desc} \mid \text{Trough} \\
 & \text{postTrough} \rightarrow \text{Asc postAsc} \mid \text{Desc postDesc} \mid \text{Peak postPeak} \mid \text{Asc} \\
 & \quad \mid \text{Desc} \mid \text{Peak} \\
 & \text{Curvature} \rightarrow \text{Curv-sharp} \mid \text{Curv-flat} \mid \text{Curv-irregular} \\
 & \text{Asc} \rightarrow \text{Asc-sharp} \mid \text{Asc-gentle} \\
 & \text{Desc} \rightarrow \text{Desc-sharp} \mid \text{Desc-gentle} \\
 & \text{Peak} \rightarrow \text{Peak-big} \mid \text{Peak-small} \\
 & \text{Trough} \rightarrow \text{Trough-big} \mid \text{Trough-small} \}
 \end{aligned} \tag{3}$$

Table 2 shows the relationship of the genetic operators analysed to determine which ones will generate valid symbolic models. Any operator should output valid individuals, that is, individuals that comply with the grammar defined in (3). Therefore,

**Table 2.** Genetic Operators

Operator Type	Operator Name	Analysis
	Tournament	After analysing the results of the experiments, these selection operators were considered not to be applicable because they have to create a huge number of generations to achieve the required convergence of the population. The tournament selection operator is the least recommendable in this case, with 500 times more generations than generational selection operator, if it converges. The next least recommendable operator for this domain would be roulette selection, with 21 times more generations, followed by the scaling selection operator, which, although it comes fairly close to the generational selection operator, will still create 4% more generations.
	Roulette	
	Scaling	
	Generational	In this case, instead of being formed at random, the first generation is composed of the population that is to produce the symbolic reference model. Therefore, the generational selection operator is the one that behaves best with respect to the number of generations produced, as the whole population is selected for crossover. This way, all the individuals have a chance of passing on their genetic load to future offspring.
Crossover	Koza	These crossover operators were not applied, because, as they are not based on a CFG, they can generate invalid individuals, and therefore the resulting symbolic reference model would not be a correct reference model.
	SCPC	
	Fair	
	Whigham	
Mutation	GBC	Both crossover operators output valid individuals, as they comply with the CFG defined in the GGGP algorithm. The Whigham crossover operator is still now in use, because of its good performance [19, 20, 21]. However, in our experiments, the GBC operator has a 3% better convergence rate, as the GBC prevents the disproportionate growth of the size of the trees representing the individuals and takes advantage of the ambiguous grammar property. This property means that there is more than one different derivation tree for the same word, the key weakness of the Whigham operator [22].
	Standard	The Standard mutation operator is usually employed by GGGP. It substitutes the subtree whose root is the mutation node for another subtree whose symbol in the root node coincides with the one in the mutation node. This constraint on matching non-terminal symbols has a negative impact on the exploration capacity of the operator when an ambiguous CFG is used. GBM (Grammar Based Mutation) overcomes this weakness.
Replacement	SSGA	De Jong applied the concept of generational replacement rate with the aim of implementing a controlled overlap between parents and offspring [12]. In this paper, a proportion $t_g$ of the population is selected for crossover. The resulting offspring will replace the worst-adapted members of the earlier population. These types of genetic algorithms, where only a few individuals are replaced, are known as SSGA (steady-state replacement genetic algorithms). This is used in the experiments run to replace the individuals of previous population.

the only operators that we were unable to apply because they could generate invalid individuals were: Koza, SCPC and Fair crossover operators.

Having eliminated these operators, we went on to experiment with the others to select the ones that led to the fastest convergence to the best possible model. As a result

of the experiments described in Table 2, the best combination of operators was: the Generational selection operator, the GBC crossover operator, the GBM mutation operator and the SSGA replacement operator.

## 6 Results and Evaluation

SYRMO was evaluated in a two-stage experimentation process. In the first stage, 500 experimental reference models were created with the aim of tuning the algorithm parameters and determining what genetic operators were best, as discussed above.

The goal of the second stage of experimentation was to evaluate the results of SYRMO. To evaluate these results, the numerical method now in use was used to generate 20 reference models. The reference models were created from populations of football, basketball and handball players. Then SYRMO was used to create the symbolic reference models from the same populations as above. When the expert in isokinetics analysed both the symbolic and numerical references models, she found that the two were very alike. In most cases, there is a perfect match between the numerical and symbolic reference models, as illustrated in Fig. 9. This figure shows a numerical reference model, represented by the extension and flexion curve, and the symbolic reference model, shown as dashed lines, which is wholly equivalent to the numerical reference model. However, some differences, which were, from the viewpoint of the isokinetics expert, not significant, were found in 5% of the cases.

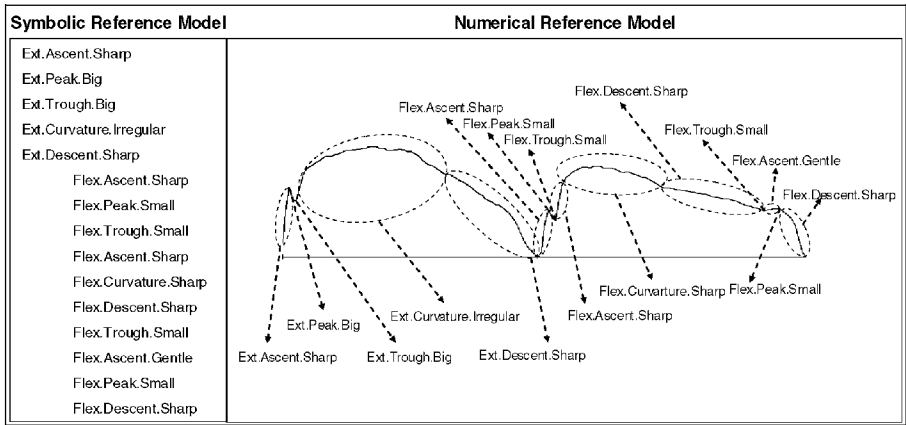


Fig. 9. Numerical Reference Model versus Symbolic Reference Model

## 7 Conclusions

This paper has presented ongoing work on the development of a comprehensive system to deal with isokinetics data, including symbolic data analysis.

Our earlier experience with numerical methods has been very positive, but experts did not have enough confidence in the system, because the information they received

did not highlight the relevant aspects of the isokinetics series in a language they found easy to understand. This is the reason that led us to introduce symbolic methods, which use the same language as our experts.

This paper has focused on the I4 project's sKDD subsystem. sKDD transforms the original numerical temporal sequences into symbolic sequences, defines a symbolic isokinetics distance (SID) that can be used to compare symbolic isokinetics sequences, and provides a method, SYRMO, for creating symbolic isokinetics reference models using grammar-guided genetic programming.

The evaluation has shown that the numerical and symbolic reference models generated from isokinetics tests on top-competition sportsmen and women are, in the expert's opinion, similar. In view of these encouraging results, we are continuing our research in the field of symbolic data analysis to build new functionalities into I4 and add symbolic injury characterization to the sKDD subsystem.

## References

- Alonso, F., Martínez, L., Pérez, A., Santamaría, A., Valente, J.P.: Symbol Extraction Method and Symbolic Distance for Analysing Medical Time Series. In: Maglaveras, N., Chouvarda, I., Koutkias, V., Brause, R. (eds.) ISBMDA 2006. LNCS (LNBI), vol. 4345, pp. 311–322. Springer, Heidelberg (2006)
- Alonso, F., Valente, J.P., Martínez, L., Montes, C.: Discovering Patterns and Reference Models in the Medical Domain of Isokinetics. In: Zurada, J.M. (ed.) *New Generations of Data Mining Applications*. IEEE Press/Wiley (2005)
- Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding Motifs in Time Series. In: *Proceedings of the 2nd Workshop on Temporal Data Mining*. At the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 53–68 (2002)
- Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: *Proceedings of 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, San Diego, California, pp. 2–11 (2003)
- Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol.* 48, 443–453 (1970)
- Luke, S.: Two Fast Tree-Creation Algorithms for Genetic Programming. *IEEE Trans. on Evolutionary Computation* 4(3), 274–283 (2000)
- Koza, J.R., Keane, M.A., Streeter, M.J., et al.: *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell (2005)
- Langdon, W.B., Poli, R.: *Foundations of Genetic Programming*. Springer, London (2001)
- Brindle, A.: *Genetic Algorithms for Function Optimization*. PhD Thesis, University of Alberta (1991)
- Baker, J.E.: Reducing Bias and Inefficiency in the Selection Algorithm. In: *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 101–111 (1987)
- Crow, J.F., Kimura, M.: *An Introduction to Population Genetics Theory*. Harper and Row, New York (1970)
- De Jong, K.A.: *Analysis of Behaviour of a class of Genetic Adaptive Systems*. PhD Thesis, University of Michigan (1975)
- Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)

- D'haesler, P.: Context Preserving Crossover in Genetic Programming. In: Proceedings of the 1994 IEEE World Congress on Computational Intelligence, Orlando, Florida, USA, vol. (1), pp. 379–407 (1994)
- Langdon, W.B.: Size Fair and Homologous Tree Genetic Programming Crossovers. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 1999, pp. 1092–1097. Morgan Kaufmann, San Francisco (1999)
- Crawford-Marks, R., Spector, L.: Size control via Size Fair Genetic Operators in the PushGP Genetic Programming System. In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, pp. 733–739 (2002)
- Whigham, P.A.: Grammatically-Based Genetic Programming. In: Rosca, J.P. (ed.) Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA, pp. 33–41 (1995)
- Couchet, J., Manrique, D., Ríos, J., Rodríguez-Patón, A.: Crossover and mutation operators for grammar-guided genetic programming. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 11(10), 943–955 (2007)
- Grosman, B., Lewin, D.R.: Adaptive genetic programming for steady-state process modeling. *Comput. Chem. Eng.* 28, 2779–2790 (2004)
- Hussain, T.S.: Attribute grammar encoding of the structure and behavior of artificial neural networks. PhD Thesis, Queen's University, Kingston, Ontario (2003)
- Rodrigues, E., Pozo, A.: Grammar-guided genetic programming and automatically defined functions. In: Proceedings of the 16th Brazilian symposium on artificial intelligence, Brazil, pp. 324–333 (2002)
- Hoai, N.X., McKay, R.I.: Is ambiguity useful or problematic for grammar guided genetic programming? A case of study. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Singapore, pp. 449–453 (2002)
- Wong, M.L., Leung, K.S.: Data mining using grammar based genetic programming and applications. Kluwer, Norwell (2002)