# Global optimization driven by genetic algorithms for disruption predictors based on APODIS architecture

G.A. Rattá[a], J. Vega[a], A. Murari[b], S. Dormido-Canto[c], R.Moreno[a] and JET Contributors*

EUROfusion Consortium, JET, Culham Science Centre, Abingdon, OX14 3DB, UK.

[a]*Laboratorio Nacional de Fusión. CIEMAT, Madrid, Spain.*
[b]*Consorzio RFX, Associazione EURATOM/ENEA per la Fusione, Padua, Italy.*
[c]*Dpto. de Informática y Automática. Universidad Nacional de Educación a Distancia. Madrid, Spain.*
** See the Appendix of F. Romanelli et al., Proc. of the 25th IAEA FEC 2014, Saint Petersburg.*
*EUROfusion Consortium, JET, Culham Science Centre, Abingdon, OX14 3DB, UK*

Since year 2010, the APODIS architecture has proven its accuracy predicting disruptions in JET tokamak. However, it has shown margins for improvements, fact indisputable after the enhanced performances achieved in posterior upgrades. In this article, a complete optimization driven by Genetic Algorithms (GA) is applied to APODIS architecture. The methodology considers all possible combination of signals, signal features, quantity of models, their characteristics and internal parameters. This global optimization aims at creating the best possible system with a reduced amount of required training data.
The results harbor no doubts about the reliability of the global optimization method, allowing to outperform the ones of previous versions: 91,77 % of predictions (89,24% with an anticipation higher than 10 ms) with a 3,55 % of false alarms. Beyond its effectiveness, it also provides the potential opportunity to develop a spectrum of future predictors using different training datasets. The analysis of how their structures reassemble and evolve in each test may help to shed some light over the inner physics of the phenomenon and to aid in the development of theoretical models to prevent disruptions in the perspective of ITER.

Keywords: Disruption prediction, Genetic Algorithms, JET, APODIS, ITER.

## 1. Introduction

Disruptions [1] can cause serious damages to the first wall and plasma facing components of tokamaks. A promising safety procedure to minimize the harm they may inflict is through their early prediction and subsequent activation of Mitigation Actions (MA) [2][3]. The prediction of disruptions is not a trivial task. The non-linear relationship of a large set of plasma parameters that may evidence a phenomenon's precursor is complex to model and the theoretical simulations are not sufficiently accurate to reliably explain or predict them [4]. These reasons have leaded the majority of disruption prediction research of the last years to data-driven systems. Up to now, the data-driven architecture that provided the best detection rates in JET has been the one named APODIS [5]. APODIS was designed to analyze the whole evolution of each shot seeking for possible disruption precursors. Based on a previous study [6], instead of using the raw measurements, a feature extraction procedure is automatically executed over them to extract disruptive-relevant signatures.

Features are vectors that contain the processed information from several signals. These feature vectors are input to three SVM classifiers [7], using a RBF Kernel (Model 3, 2 and 1 in Figure 1) built to work in parallel and simultaneously over three consecutive time windows.

In case of just one model, the decision would be straightforward. APODIS architecture, instead, makes three simultaneous predictions and they may disagree. To determine whether these 3 predictions are recognizing a disruption precursor or not, a second layer with a SVM classifier (Decision Function) is used.

The Kernel values [7] ( and C for RBF and C for the Linear) were set via series of systematic scans of a reasonable range of parameters. The study determined that, for that specific database, the optimal number of models in the first layer should be 3. Since its first introduction in 2010, this system has been subjected to upgrades in 2012 [8] and 2013 [9] (the latter installed online in the JET real time data network). Both upgrades primarily consisted on revising the most adequate set of signal values (and signal features) to analyze in order to predict disruptions. These upgrades have proven that notably better results can be reached with a careful selection of these variables. Nevertheless, revisions should have also included the reassembling of the architecture: a new combination of signals could fit better if, for example, the predictor has four models in the first layer instead of three.

The goal of this study is to perform this general optimization in affordable computational times.

*author's email: firstname.lastname@some.mail.server*

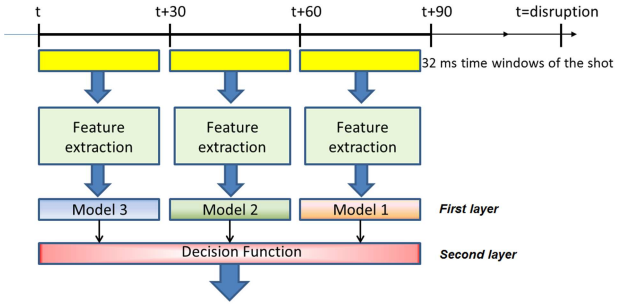


Figure 2.Modular scheme of the APODIS architecture.

## 2. Database and signals

The database was collected from JET with the ITER-like wall and includes 9 signals: 1) plasma current (IPLA) [A]; 2) mode lock amplitude (ML) [T]; 3) plasma internal inductance [Wb/A]; 4) line averaged density (ne) [m−3]; 5) stored diamagnetic energy time derivative [W]; 6) total radiated power (Prad) [W]; 7) total input power [W]; 8) poloidal beta; toroidal magnetic field [T]. These signals are processed to extract from them disruptive-relevant features vectors in a similar procedure as the one applied in the Online version of APODIS [9][10].

Feature vectors contain 2 values per signal. One feature is the standard deviation of the Discrete Fourier Transform (discarding the CC component) computed over the past 32 ms of each signal [6]. The other feature is its amplitude value. This second feature differs of the ones used previous APODIS versions (i.e. the mean value of the amplitude calculated for that previous 30 ms). This change aims at looking for fast variations that could be smoothed if they are averaged. All signals have been processed following a strict real-time simulation, which means that it can be exactly reproduced under online conditions [10][11].

Two sets of discharges have been gathered. D1 contains 1178 shots (144 disruptives) produced during the period between January 2012 and July 2012. This dataset has been used to train the prediction system. An independent group of 467 discharges (158 disruptives) from July 2013 to September 2013 has been saved for testing, simulating real-time operation, the developed predictor performance.

Intentionally produced disruptions have been discarded from the database. Since they are designed and forced to occur at a predefined instant of a discharge, they do not evolve as the unintentional ones and their analysis may mislead a machine learning system. The prediction of intentionally provoked disruptions is not a pursued goal in this study.

## 3. The computational problem

The proposed global optimization must include all the parameters involved in the predictor. These parameters are: 1) 4 possible quantities of models in the first layer; 2) 4 possible widths (in ms) of the models; 3) 8 possible values for $\gamma$; 4) 8 possible values for $C$; 5) signal (and signal features) to be considered: 18 values. They complete a total of 42 values. An exhaustive analysis (creating and testing each possible abovementioned combination without considering permutations) is defined by:

$$\sum_{i=1}^{n} C_i^n$$

$$C_i^n = \frac{n!}{(n-i)!i!} \qquad (1)$$

where:

$n$=number of possible values= 42 and $i$=possible groupings=1,2,3,…,42

Exploring all this possibilities, considering that each one requires an estimated computational time of ~3 seconds, would take **418380 years**.

Genetic Algorithms (GA) can be applied to solve the problem with elegant efficiency.

## 4. Genetic Algorithms

### 4.1. Introduction

GA is a set of computer methods inspired by biological evolution and aimed to perform user-defined tasks, normally for the optimization of complex numerical problems.

In nature, better adapted individuals have higher chances to survive and breed descendants. In this context, adaptation means to reach, among others, the objectives of survival and reproduction. The adapted individuals are able to transfer to the next generation their genetic material and their offspring will inherit a combination of "well-adapted" parents' genes. Oppositely, individuals unable to survive and reproduce will not pass their characteristics and therefore their configuration is destined to extinction.

GA uses this basic principle. Given a problem, a population of possible solutions/individuals is created. To measure which individuals are better adapted, a metric called Fitness Function (FF), that scores individuals performances, is applied. According the FF score, higher possibility to mate and have descendants is assigned to individuals/solutions with better values. Descendants are created as a combination of parents characteristics/genes. Finally, and since descendants are a combination of promising genes, it is expected that newer generations will outperform the former ones. The procedure can be summarized in few steps:

1- Creation of a population of individuals (each individual represents a possible solution to a given problem).

2- Evaluation of each individual of the population according the objectives of the problem. This requires to have defined a metric to test how well each individual is solving the problem (i.e. the FF).

3- Selection of parents (a higher probability to be chosen as parent is assigned to those individuals with higher FF values).

4- Creation of children as a combination of parents' genes (using genetic operators as crossover, mutation, reproductor).

5- Unless an ending condition is satisfied, iterate from step 2, where the new population (created in step 4) is evaluated.

The power of GA to reduce computational resides in these basic principles. Instead of exploring all the possible combinations, the most promising ones are chosen and crossed to create new solutions prone to outperform the former ones.

Notice that the whole procedure requires encoding each solution/individual as a set of interchangeable characteristics/genes (to execute step 4). This is an essential part to achieve good results with GA.

### 4.2. Codification in nature and GA equivalence

Every living organism carries a digital code (DNA or RNA) that encloses all the information to create (even clone) such individual. In case an individual mates, its DNA is crossed with the partners' DNA melting into a new set of instructions to create offspring. One way to encode an optimization problem using GA is to define a string as a computational imitation of the DNA coiled in chromosomes. There, genes are emulated by bits (see Figure 2).

In the case of this study, the structure is encoded, as schematized in Figure 3. In the strings NoM defines the Number of Models in the first layer (see Figure 1). WoM is the gap (in ms) between the Models in the first layer.

The inclusion (or not) of any signal (light blue) or signal feature (std(DFT), in green) into the predictor's design is determined by the nonzero value (or zero) of its corresponding bit.

The codification of NoM has been set as:{'00'→1}; {'01'→2}; {'10'→3}; {'11'→4}; for WoM: {'00'→8}; {'01'→16}; {'10'→32}; {'11'→64}; for $\gamma$ {'000'→0.001}; {'001'→0.005}; {'010'→0.01}; {'011'→0.05}; {'100'→0.1};{'101'→0.5}; {'110'→1}; {'111'→10}; and for *C*: {'000'→ 0.01}; {'001'→ 0.1}; {'010'→1}; {'011'→5}; {'100'→10}; {'101'→50}; {'110'→100}; {'111'→1000}.

The example string depicted in Figure 3 defines the instructions of a predictor that as plasma feature would use: 1) the std(DFT(32 ms)) of the Plasma Current (PC) and 2) the amplitude of the Mode Lock signal (ML). It would have 4 models in the first layer (NoM={'11'→4}), each one of them separated 16 ms (WoM={'01'→16}), with a $\gamma$ of 10 ({'111'→10}) and a *C* value of 1 ({'010'→1}).

### 4.3 Creation of the first population

Each individual of the population is initially created as a string of randomly selected ones and zeros. One parameter to be set is the number of individuals per generation (size of the population). Even if some publications are meant to estimate an appropriate quantity [12], this value is usually problem-dependent.

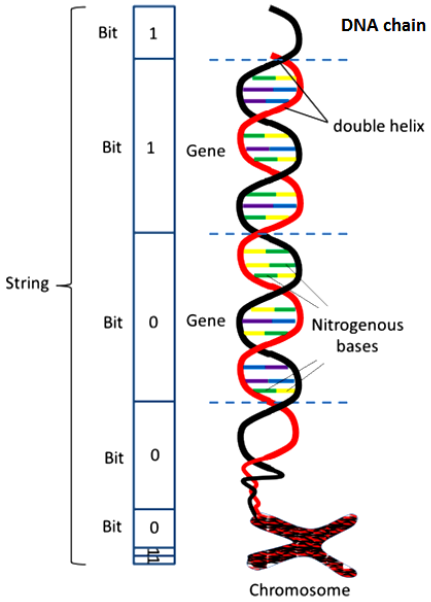


Figure 2. Simplified illustration of parallelisms between DNA and its GA codifications.

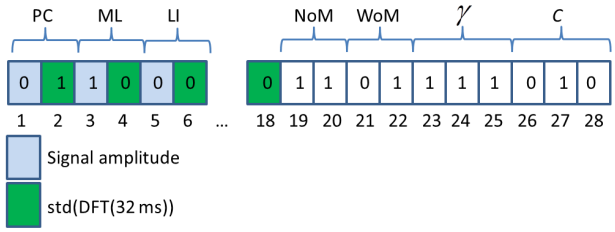


Figure 3. An individual/predictor is defined by its codification.

In this case the chosen quantity was 28 individuals (equal to the length of each string). In this step, 28 strings with 28 randomly distributed ones and zeros are created.

### 4.4. Training predictor candidates

Each individual of the population is just a set of 28 strings randomly filled with ones and zeros. Each string is not a disruption predictor as a DNA is not an animal. They are the instructions to create one.

A Building Unit (BU) is a necessary program aimed at transforming the instructions encoded in the strings/individuals into a predictor. The BU, then, trains a predictor candidate by following the codified instructions. The training procedure is analogous to the one detailed in previous APODIS versions [5][9][8]. After this procedure each individual/string became a trained predictor candidate able to be validated in real-time simulations.

### 4.5. Validation of candidates using the Fitness Function

The same set of discharges (D1) was used for training and validation. A fundamental problem arises in these cases. A ML-based system may learn "by heart" what have happened before (over-fitting) instead of creating models with generalization capabilities. In order to avoid over-fitting, each individual is twice trained (BU of previous section) and tested in independent runs.

Different training and testing samples are selected each time (2-fold cross-validation method). This method is performed twice, obtaining 4 results per predictor candidate. The results are averaged to obtain the final FF value per candidate. The FF consist of the results of the candidates during the validation according a previously established score system: for each predictor candidate 5 points were assigned per disruption predicted between 10 ms Before the Disruption (BF) and 1 second BF (to avoid premature alarms); 2 points in case the system detects a disruption but with less than 10 ms of anticipation; 3 points in case, correctly, no alarm is triggered in a non-disruptive shot. This score system steers the course of the evolution towards the type of predictor aimed to obtain. Notice that it can be easily modified to obtain different types of predictors (e.g. increasing the points per non-triggering alarms in safe shots would lead to more cautious systems, with lower rates of false alarms but higher rates of missed alarms).

After this evaluation a FF each individual has attached their FF score. The parent selection method that uses these scores was the one proposed by Baker [13]:

1. A list with the 28 candidates is sorted according their FF values.

2. FF values are normalized between 0 (lower) and 1 (higher).

3. A random number between 0 and 1 is chosen (e.g. 0.65).

4. Candidates with normalized FF higher than the random number are selected as parents.

In the case that the random value is different from 0, just a fraction of the candidates are chosen. It is necessary to select 28 parents in this method. In these cases, above mentioned steps 3 and 4 are repeated until the number of selected candidates is 28.Notice that from these 28 parents, some candidates may be copied several times.

### 4.6 Cross-over and mutation

The objective of parents' selection is to cross their promising genes/bits (the better their FF, the higher the chances to be selected) to create children.

From the 28 parents, 14 pairs are selected randomly. Their genes/bits are mixed using the 2 point cross-over operation. As it is schematized in Figure 4, two random points are chosen from parents' codes. The sections delimited by these points are interchanged. As consequence, two children are created.

Also, mutation possibility has been implemented. It consists of flipping a bit's value in children. This operation is useful to increase the diversity of the gene pool and to avoid local minimal. Following the De Jong's criteria [14]the mutation probability was set as a 0.036%.

In this stage, a new population of 28 individuals (children) is created. The first iteration began with a random assignation of string values. However, in every posterior generation, the replacing populations are driven by GA by mixing the most fitted configurations.

The whole process is repeated (from Subsection 4.4) until an ending condition is fulfilled (50 iterations, enough to achieve acceptable results in this case).

The candidate with the higher FF of all generations was selected as final predictor (GA-APODIS). The characteristics of GA-APODIS resulting in this optimization were: 2 models with a WoM of 32 ms in the first layer; value equal to 0,001 and a C value of 1;using as signals the ML and Prad and as signal features the ML, IPLA and ne. The whole procedure to attain GA-APODIS required ~4 hours of computational time.

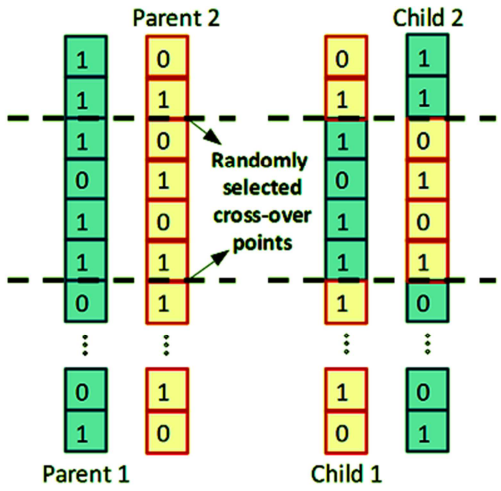


Figure 4.Schematization of the 2 points cross-over operation.

## 5. Application in simulated real-time and results

APODIS-Online (7 signals and 7 signal features) and the GA-APODIS were trained using D1. The testing results displayed in this section were obtained following a strict real-time simulation, inspecting the evolution of each discharge contained in the independent testing dataset (see section 2). The summary of the tests is shown in Table 1. There, 'ideal' refers to the alarms triggered at least 10 ms before the disruption but not earlier than 1 second; 'correct' are all the ones activated at least with 10 ms of anticipation; 'late' are the ones detected with less than 10 ms in advance; 'total' are the 'correct' plus 'late' alarms; 'premature' are alarms flagged too early (i.e. > 1 second before the disruption occurrence) and 'false' are alarms activated in discharges that did not end in a disruption.

Results reveal that GA-APODIS considerably outperformed APODIS-Online (trained with the same database). However, they are similar to the ones obtained with the APODIS-Online version (trained with a wide database of 8407 discharges). Regarding the distribution of the predictions, it is interesting to notice that the equilibrium of the GA-APODIS follows, as expected, the guidelines established by the scoring system. A considerable reduction of the premature alarms ~3% was achieved according to the reward of 5 points (only if the prediction is not premature). The consequence was a

~2% higher amount of false alarms in comparison with APODIS-Online configurations.

## 6. Summary and discussion

A global optimization methodology for disruption predictors based on APODIS architecture has been developed. It provides considerable improvements in terms of prediction rates in case of using medium size databases for the training. The GA-APODIS predictor, in this case and for this particular database required 4 hours of computational time, which includes the 50 iterations of the programmed GA. A scoring system, easily modifiable, steered the course of the evolution towards the desired predictor. Triggering faster alarms (or any other particular expected response of the predictor) can be fostered just by rewarding with higher scores each behavior in the optimization. This flexibility is a formidable tool to be considered and should be exploited in future works.

| | Ideal [%] | Correct [%] | Late [%] | Total [%] | Premature [%] | False [%] |
|---|---|---|---|---|---|---|
| APODIS-Online (trained with a wide database) | 81 | 88,6 | 2,53 | 91,13 | 7,5 | 1,62 |
| **APODIS-Online (trained with database D1)** | **66,45** | **73,41** | **12,65** | **86,076** | **6,96** | **1,31** |
| **APODIS-GA (trained with database D1)** | **86,08** | **89.24** | **2,53** | **91,77** | **3,16** | **3,55** |

Table 1. Summary of results

In the perspective of ITER, the unavailability of disruption databases at the beginning of its operation can be an issue for the creation of data-driven systems to avoid disruptions. The introduced procedure should, in the near future, be put under thorough testing by training several predictors with different sets of discharges. The reassembling in the architecture, the chosen set of signals and kernel parameters under each set of discharges could provide key hints to undercover the physics of the phenomenon and to aid a better understanding of it from a theoretical perspective. The predictor described in this paper could also be complemented with other adaptive approaches as the published in [15] and extended to the problem of disruption avoidance.

## Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

This work was partially funded by the Spanish Ministry of Economy and Competitiveness under the Projects No ENE2012-38970-C04-01.

## References

[1] F.C. Schuller. 1995. "Disruption in tokamaks". Plasma Phys. Control. Fusion 37. A135–62.

[2] M. Lehnen et al 2013 Impact and mitigation of disruptions with the ITER-like wall in JET. Nucl. Fusion 53 093007 doi:10.1088/0029-5515/53/9/093007

[3] L.R. Baylor et al. 2009. Pellet fuelling, ELM pacing and disruption mitigation technology development for ITER. Nucl. Fusion 49 085013 doi:10.1088/0029-5515/49/8/085013

[4] A.H. Boozer. 2012. "Theory of tokamak disruptions". Physics of Plasmas. 19, 058101.

[5] G. A. Rattá, et al. "An Advanced Disruption Predictor for JET tested in a simulated Real Time Environment" Nuclear Fusion. 50 (2010) 025005 (10pp).

[6] G.A. Rattá, et al. 2008. "Feature extraction for improved disruption prediction analysis at JET". Rev. Sci. Instrum. 79 10F328

[7] Cortes C. and Vapnik V. 1995. "Support-vector networks". Mach. Learn. 20 273–97

[8] G.A. Rattá, J. Vega, A. Murari, JET-EFDA Contributors. "Improved feature selection based on genetic algorithms for real time disruption prediction on JET". Fusion Engineering and Design (2012) http://dx.doi.org/10.1016/j.fusengdes.2012.07.002

[9] J. Vega et al. "Results of the JET real-time disruption predictor in the ITER-like wall campaigns". Fusion Engineering and Design 88 (2013) 1228-1231.

[10] J.M. Lopez, et al. 2014. Implementation of the Disruption Predictor APODIS in JET's Real-Time Network Using the MARTe Framework. Nuclear Science, IEEE Transactions on (Volume:61 , Issue: 2 ) pp 741 - 744 ISSN 0018-9499 10.1109/TNS.2014.2309254

[11] López J.M. 2012 Implementation of the disruption predictor APODIS in JET real-time network using the MARTe framework Proc. 18th IEEE-NPSS Real Time Conf. doi:10.1109/RTC.2012.6418168

[12] J.T. Alander. On optimal population size of genetic algorithms. Proceedings CompEuro, Computer Systems and Software Engineering, 6th Annual European Computer Conference, 1992, pp. 65–70.

[13] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, 1987, pp. 14–21

[14] K. A. De Jong, W.M. Spears. "An analysis of the interacting roles of population size and crossover in genetic algorithms". Parallel Problem Solving from Nature. Lecture Notes in Computer Science Volume 496, 1991, pp 38-47.

[15] A.Murari et al,Nucl. Fusion 53 (2013) 033006 (9pp)