Software para análisis de actividad neuronal en tiempo real durante neurocirugía

Villafañe Sebastián Omar

Ingeniería Biomédica

Escuela de Ciencia y Tecnología

UNSAM

Director de Carrera: La Mura Guillermo

Profesor a cargo: Salvucci Fernando

Supervisora: Andres Daniela

Resumen

La localización de target anatómico para la implantación de electrodos cerebrales es un punto crítico en la cirugía de estimulación cerebral, la que se utiliza para enfermedad de Parkinson y otras patologías. El target anatómico se define combinando imágenes médicas con técnicas de análisis de señales de actividad neuronal (microregistro neuronal). Actualmente existe la necesidad de poder realizar análisis automatizados de microregistro neuronal en tiempo real y no depender de las habilidades de un experto para poder confirmar un target anatómico. Se propone como solución un software que sea fácil de utilizar, sencillo, intuitivo, veloz y a la vez con la posibilidad de modificar ciertos parámetros para adaptarlo a diferentes tipos de análisis. La aplicación cuenta con un proceso de carga y presentación de datos en pantalla y con la detección de spikes (potenciales de acción) mediante umbrales modificables por el usuario. Una vez detectados los spikes se realiza un análisis wavelet de 10 dimensiones para caracterizarlos. Posteriormente se realiza spike sorting utilizando un concepto de inteligencia artificial basado en la biología evolutiva: un algoritmo genético. Si bien los algoritmos genéticos son poderosos a la hora de realizar clustering no supervisado (clasificación de conjuntos de datos desconocidos), presentan el problema general conocido como *overfitting*, por el que tienden a encontrar mayor número de *clusters* o grupos de los que en realidad existen. Una solución original para el problema común de overfitting en los procesos de spike sorting es propuesta en este trabajo. Con los clusters-neuronas ya aislados, es posible aplicar un algoritmo de detección de target que se basa en un análisis de los intervalos entre spikes y la organización temporal del incremento de los mismos, logrando distinguir entre distintas estructuras anatómicas. El software presenta este resultado como una probabilidad, permitiendo validar un blanco quirúrgico. Posteriores desarrollos podrían integrar la aplicación con un sistema de adquisición de datos propietario y realizar versiones adaptadas para plataformas móviles. Aunque el mayor desafío de este software es el tiempo de procesamiento requerido, se puede afirmar que se ha logrado buena performance para el soporte en la toma de decisiones en tiempo real durante neurocirugías.

Introducción

La enfermedad de Parkinson es un desorden neurológico crónico y progresivo (Schrag, Jahanshahi et al. 2000, Jankovic 2008). Eso significa que sus síntomas empeoran con el tiempo. Actualmente se desconocen su causa y su cura, por lo que su tratamiento se basa en el control de los síntomas de la enfermedad (Jankovic 2008, Goetz 2011). Más de 10 millones de personas viven con la enfermedad de Parkinson en el mundo (Parkinson's Disease Foundation 2017, Parkinson's Disease Foundation 2017). Aunque su incidencia se incrementa con la edad, aproximadamente el 4% de las personas con esta patología son diagnosticadas antes de los 50 años (Parkinson's Disease Foundation 2017).

Al presente, el tratamiento de la enfermedad de Parkinson ofrece dos alternativas (Frades-Payo, Forjaz et al. 2009) . La primera alternativa consiste en el tratamiento farmacológico y la segunda en la intervención quirúrgica (Martinez-Martin and Deuschl 2007) . Si bien la terapia farmacológica permite un control de los síntomas de la enfermedad, existe una alta probabilidad de que con los años se presenten efectos adversos como la disminución de la duración del efecto de las drogas o movimientos involuntarios (disquinesias) (Fahn 1996, Obeso, Rodriguez-Oroz et al. 1999, Rodriguez-Oroz, Obeso et al. 2005, Juri and Chaná 2006). En el caso del tratamiento quirúrgico, las opciones son la cirugía ablativa y la cirugía de estimulación cerebral profunda (DBS) (Guridi, Rodriguez-Oroz et al. 2004, Rodriguez-Oroz, Obeso et al. 2005, Sandoval, Jiménez et al. 2010). La cirugía ablativa es una intervención donde se destruye una pequeña parte del cerebro con termolesión. En el año 1940 se realizó lo que se conoce como la primera intervención a los ganglios de la base (considerados en esa época el centro de la conciencia) (Meyers 1942). El blanco anatómico a intervenir fue variando a través de los años gracias a los trabajos de distintos especialistas hasta la aparición de nuevas terapias farmacológicas. A finales de la década del 50 se comienza a considerar que el neurotransmisor denominado *dopamina* podría estar asociado con la sintomatología de la enfermedad de Parkinson (Hornykiewicz 2006, Björklund and Dunnett 2007, Goetz 2011). En los años 60 comenzaron a realizarse pruebas de tratamiento con Levodopa (L-dopa), la que se impuso debido a su alta eficiencia en el control de los síntomas de la enfermedad (Barbeau 1969). Años después la cirugía vuelve a ser considerada como un método terapéutico altamente eficaz, debido a la elevada frecuencia de efectos adversos en pacientes bajo terapia farmacológica (hasta 90%) (Rodriguez-Oroz, Obeso et al. 2005, Goetz 2011).

Hoy en día la ablación de ciertos núcleos es útil para personas con contraindicaciones en la implantación de dispositivos o países con limitados recursos económicos (Machado, Rezai et al. 2006). Sin embargo, la opción quirúrgica electiva en la mayoría de los casos es la estimulación cerebral profunda. La misma consiste en la estimulación eléctrica continua de un blanco anatómico a través de electrodos implantados crónicamente. Estos electrodos se encuentran conectados a un dispositivo estimulador que suele ser programable en amplitud, pulso y frecuencia. En el caso de la enfermedad de Parkinson, los blancos anatómicos por excelencia son el núcleo subtalámico (STN) y el globo pálido interno (GPi) (Kumar, Lozano et al. 1998, Burchiel, Anderson et al. 1999,

Anderson, Burchiel et al. 2005). La estimulación de alta frecuencia de estos núcleos ha demostrado en varias oportunidades su efectividad y seguridad (Kumar, Lozano et al. 1998, Trépanier, Kumar et al. 2000, Benabid 2003). Una de las principales ventajas de la estimulación cerebral profunda sobre la ablación es su reversibilidad, ya que la cirugía ablativa genera un daño irreparable sobre blancos anatómicos, permitiendo en muchos casos controlar los síntomas de la enfermedad bajo el riesgo de producir efectos adversos (Benabid, Pollak et al. 1987, Schuurman, Bosch et al. 2000, Rodriguez-Oroz, Obeso et al. 2005).

La localización del sitio de estimulación es un punto crítico en la cirugía (Benabid 2003, Falkenberg, McNames et al. 2006). El procedimiento quirúrgico consta de una serie de pasos. En primer lugar, se realiza la pre identificación del blanco neural con un atlas cerebral basado en los síntomas que presenta el paciente. Para reconfirmar la ubicación definida, se pueden agregar uno o varios métodos complementarios de identificación del blanco anatómico: MRI (imágenes por resonancia magnética), TC (tomografía computada), ventriculografía y microregistro neuronal. (Benabid 2003, Machado, Rezai et al. 2006). Para poder utilizar un sistema de imágenes como resonancia magnética o tomografía computada, es necesario agregar marcadores o localizadores antes de realizar la obtención de la imagen para poder luego referenciar la posición en el volumen correspondiente (Bejjani, Dormont et al. 2000, Machado, Rezai et al. 2006). Este procedimiento, permite disminuir el margen de error a la hora de realizar la intervención. Sin embargo, la resolución típica de los sistemas de adquisición de imágenes, combinada con los errores sistemáticos, producen que la precisión anatómica final del procedimiento no sea suficiente para la ubicación de la región crítica a estimular (Guridi, Gorospe et al. 1999) .

Una técnica complementaria al diagnóstico por imágenes es la técnica de microregistro neuronal (microelectrode recording, MER) (Lozano, Hutchison et al. 1996). Esta técnica consiste en la implantación de microelectrodos a través de una cánula, utilizando para insertarlos un sistema motorizado con precisión sub-milimétrica. Generalmente se suelen utilizar microelectrodos de platino/iridio (Pt/Ir 80/20%), con impedancias nominales similares a 0.8-1.2 MΩ. El electrodo se encuentra conectado a un sistema de adquisición de datos, que realiza preamplificación, acondicionamiento y grabación de los registros la frecuencia establecida (generalmente en el orden de los 50.000 Hz) para procesarlos posteriormente offline con una sistema informático (Machado, Rezai et al. 2006, Andres, Cerquetti et al. 2011). Esta técnica aporta mayor precisión para determinar el sitio anatómico y permite realizar una validación fisiológica de la ubicación del electrodo en profundidad (Machado, Rezai et al. 2006). Aunque su efectividad para la localización del blanco anatómico se encuentra aceptada, la técnica no se encuentra aún automatizada ni estandarizada. Las características del equipamiento y el procedimiento utilizado, difiere en cada caso según sea el grupo de especialistas que interviene, del centro en el cual se realiza la intervención y de las condiciones del sistema de salud al que se está sometido (Machado, Rezai et al. 2006).

En la figura 1 se presenta un corte coronal del encéfalo y las estructuras en ella identificables.

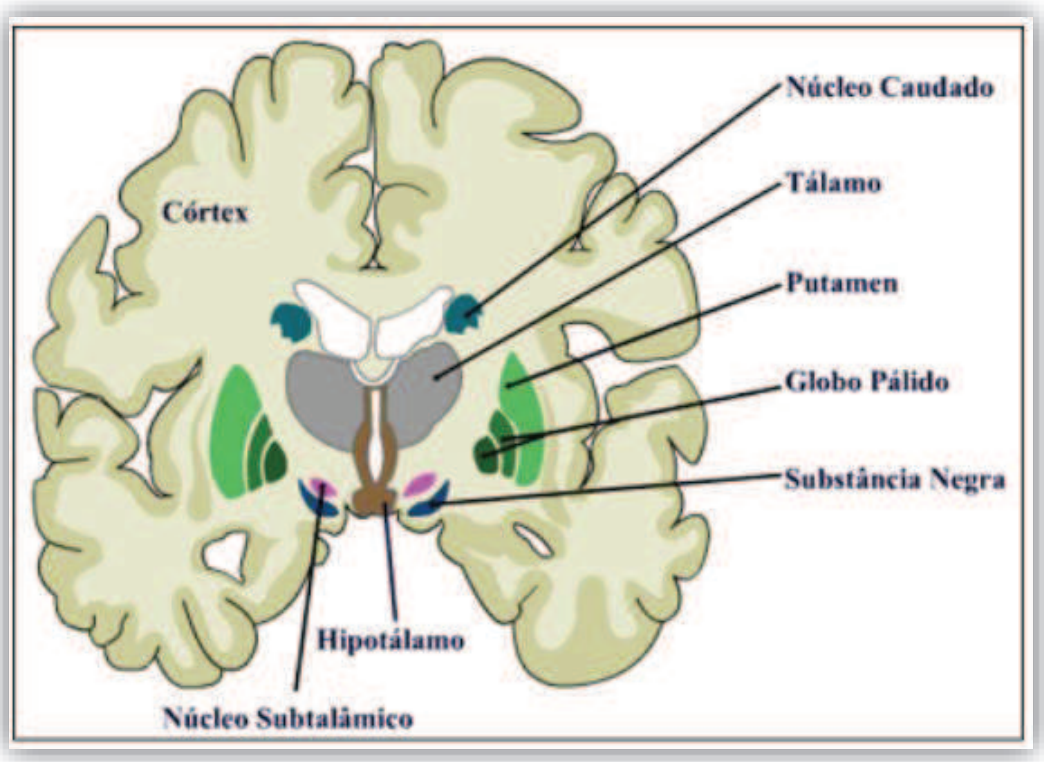


Figura 1: Corte coronal del encéfalo, dónde se pueden apreciar los ganglios basales. Estas estructuras subcorticales son las que se debe identificar para lograr su estimulación eléctrica mediante electrodos profundos,

(Antroporama.net)

Dependiendo del ángulo elegido para introducir un electrodo en el cerebro se obtendrán distintas trayectorias para intentar acceder a una región o un blanco en particular. Una vez seleccionada la trayectoria es necesario establecer la profundidad a la cual debe llegar el electrodo para alcanzar el blanco de estimulación. Esta tarea es ejecutada por un especialista que determina si el electrodo se encuentra bien posicionado según la señal de MER. El resultado, es altamente dependiente de la experiencia del operador. La figura 2 ilustra dos trayectorias diferentes (representadas por las líneas rectas) a través del núcleo subtalámico, un blanco quirúrgico común. Según el recorrido elegido y el error al cual estemos sometidos en el posicionamiento y la profundidad, variarán las posibles estructuras que podremos encontrar o atravesar buscando el blanco quirúrgico. La tarea del especialista consiste en diferenciar el MER de cada estructura con las adyacentes, para determinar la posición del electrodo.

La importancia de realizar una aplicación de análisis automático de MER es que permitirá al especialista contar con la información probabilística necesaria para decidir si el electrodo se encuentra en el objetivo anatómico esperado o no. De esta manera se acelera la curva de aprendizaje y se posibilita la transferencia de la técnica quirúrgica a centros menos especializados. Dado que este proceso será automático, es necesario contar con algoritmos desatendidos que, mediante parámetros configurables, puedan aplicar distintos criterios. La señal MER se encuentra formada por *spikes* (potenciales de acción producidos por la actividad neuronal en el sitio anatómico blanco) y

actividad neuronal de fondo (producida por neuronas que se encuentran más lejos del electrodo de registro). Para lograr el análisis de esta señal es necesario en primer lugar detectar, caracterizar y clasificar los spikes. El proceso de detección y caracterización de spikes que realiza nuestra implementación está basado en una publicación ampliamente reconocida y aceptada por la comunidad científica denominada *"Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering*" del año 2004 (Quiroga, Nadasdy et al. 2004). Una vez detectados los spikes debe llevarse a cabo su clasificación o *clustering*, lo que se conoce como *spike sorting*, con el objetivo de aislar series de actividad de neuronas individuales (Lewicki 1998). Este proceso es fundamental en el funcionamiento de la aplicación, ya que se encarga de agrupar los spikes en distintas neuronas. El agrupamiento de los spikes no es un tema trivial; existen muchos métodos que se basan en distintas condiciones para decidir a qué grupo o neurona pertenece cada spike (Lewicki 1998, Letelier and Weber 2000, Maulik and Bandyopadhyay 2000, Shoham, Fellows et al. 2003). Desde el punto de vista del análisis de datos, el problema consiste en la clasificación no-supervisada de un conjunto de datos pertenecientes a un número de subgrupos desconocidos.

El *spike sorting* es un proceso es de vital importancia. Este algoritmo debe ser capaz de determinar cuántas neuronas han sido las generadoras del registro MER y agruparlas/identificarlas en clusters-neuronas. Esta solución debe ser independiente del valor que el usuario le ingrese al software como número de clusters (K-máximo) para intentar agrupar los spikes. Un mal agrupamiento podría determinar el fracaso de la detección del proceso de target anatómico, ya que al mezclar los spikes de distintos tejidos cerebrales, generará patrones y calculará intervalos temporales que no se corresponderán con ninguna sección anatómica característica. En este trabajo se desarrolló e implementó un procedimiento de spike sorting basado en un algoritmo genético (Maulik and Bandyopadhyay 2000).

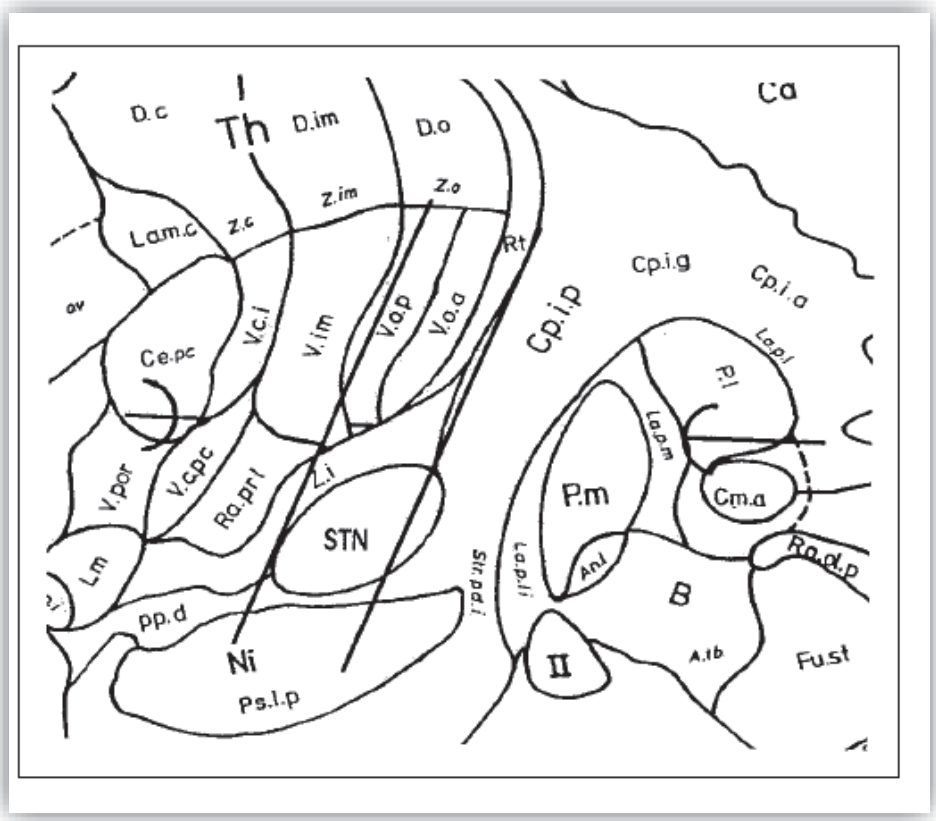


Figura 2: Problemática actual en la localización del blanco anatómico. Vista sagital del atlas de Schaltenbrand and Wahren; las dos líneas rectas presentan dos trayectorias diferentes. Según el recorrido elegido y el error al cual estemos sometidos en el posicionamiento y la profundidad, variarán las posibles estructuras que podremos encontrar o atravesar buscando el blanco quirúrgico.(Guridi, Rodriguez-Oroz et al. 2000)

Objetivo

El objetivo de este trabajo es desarrollar un software para el análisis en tiempo real de señales de microregistro neuronal durante neurocirugía funcional para la localización anatómica de blanco de estimulación. El desarrollo implica también el diseño de otras funcionalidades que permitirán alcanzar el objetivo anteriormente citado.

i. Diseñar una aplicación que permita llevar a cabo las siguientes tareas:

- Detectar potenciales de acción (spikes) en una señal de microregistro neuronal.
- Clasificar los spikes para aislar actividad de células únicas.
- Aplicar un algoritmo de análisis para caracterizar propiedades no lineales de la neurona.
- Localizar anatómicamente la neurona en términos de probabilidad, como soporte para la toma de decisiones por parte del especialista usuario.

ii. Características de la aplicación:

a. Contar con una interfaz amigable.

b. Funcionamiento intuitivo para usuarios no técnicos.

c. Velocidad de ejecución óptima para utilizarse durante una intervención neuroquirúrgica y poder asistir en la toma rápida de decisiones en situaciones médicas críticas.

d. Funcionamiento sin conexión de red para poder utilizarse en un entorno quirúrgico.

e. Flexibilidad para la modificación de parámetros de análisis por parte del usuario.

Desarrollo de software

Front-end

Para desarrollar esta aplicación fue utilizado íntegramente el lenguaje Visual C#. La versión final del proyecto contiene más de 5000 líneas de código fuente. El entorno de desarrollo utilizado fue Visual Studio 2017 sobre un sistema operativo Microsoft Windows 10 Home x64. El software fue basado en el funcionamiento de los *Windows Forms*, logrando como resultado final una aplicación intuitiva para la mayoría de los usuarios y de baja complejidad de uso. Se decidió realizar el front-end con sólo 5 botones para simplificar su ejecución y que la aplicación los habilitase según se superaran las etapas de procesamiento (figura 3). Además, se agregó una barra de progreso y etiquetas de estado para indicar la fase en la que se encuentra el proceso y resultados parciales de procesamiento. Para poder presentar resultados al usuario se utilizaron 4 ventanas gráficas que cambian su contenido según la etapa en la que se encuentra el algoritmo. Para el ajuste de parámetros se implementó un formulario de configuración que permite modificar parámetros avanzados u optar por cargar valores por defecto (*default*) que se encuentran precargados en el código fuente (figura 4). Para el diseño de la aplicación se eligió un look & feel basado en las paletas de colores habitualmente utilizadas por Microsoft ya que se intentó crear un entorno de aspecto conocido para el usuario promedio.

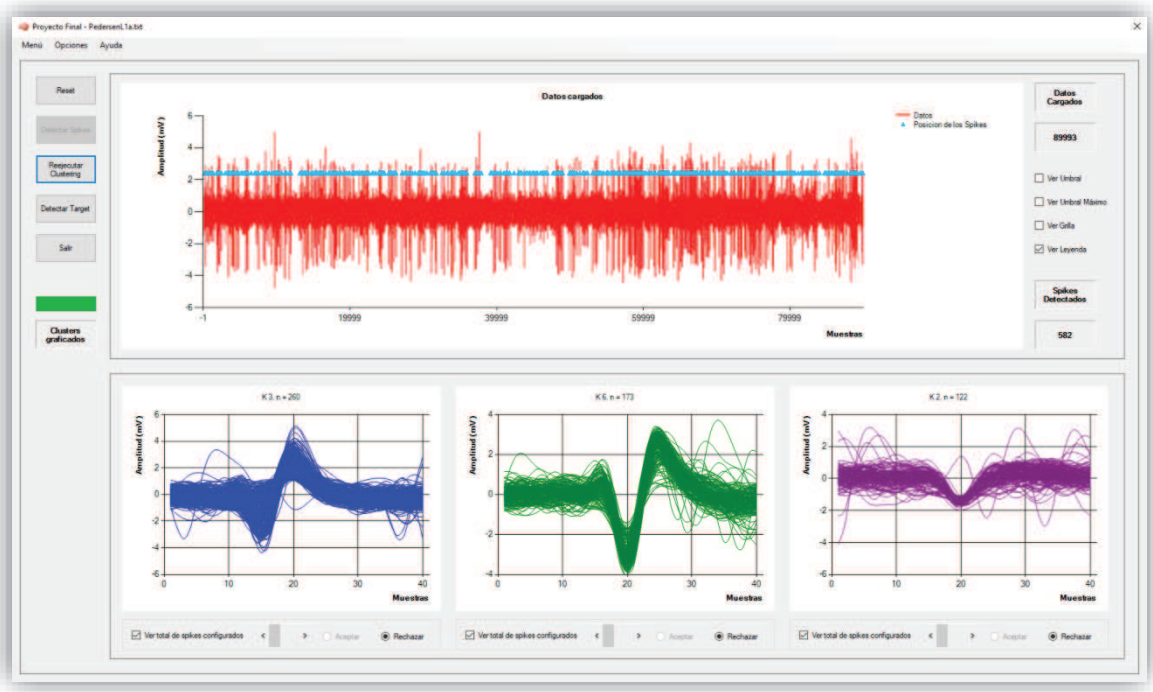


Figura 3: Aspecto del Front-end de la aplicación en pleno procesamiento de una señal. Desarrollada en Visual C# y basada en Windows Forms. La sección superior presenta una señal modelo de microregistro neuronal con los spikes detectados. Se pueden apreciar los 5 botones de la aplicación y los checkbox que permiten mostrar umbrales, grilla y la leyenda. Luego de finalizar la ejecución del Clustering Genético, se presentan en la parte inferior los 3 clusters con mayor cantidad de spikes agrupados. Se pueden visualizar los potenciales de acción individualmente con la barra de navegación o todos juntos para finalmente decidir si se termina aceptando o rechazando el clúster en cuestión. El rechazo implica que no participará del algoritmo de Detección de target.

El software ofrece la posibilidad de cambiar los parámetros más destacados de sus algoritmos principales. A través del menú *Opciones* es posible acceder al formulario de configuración de los mismos, tal como se muestra en la figura 4.

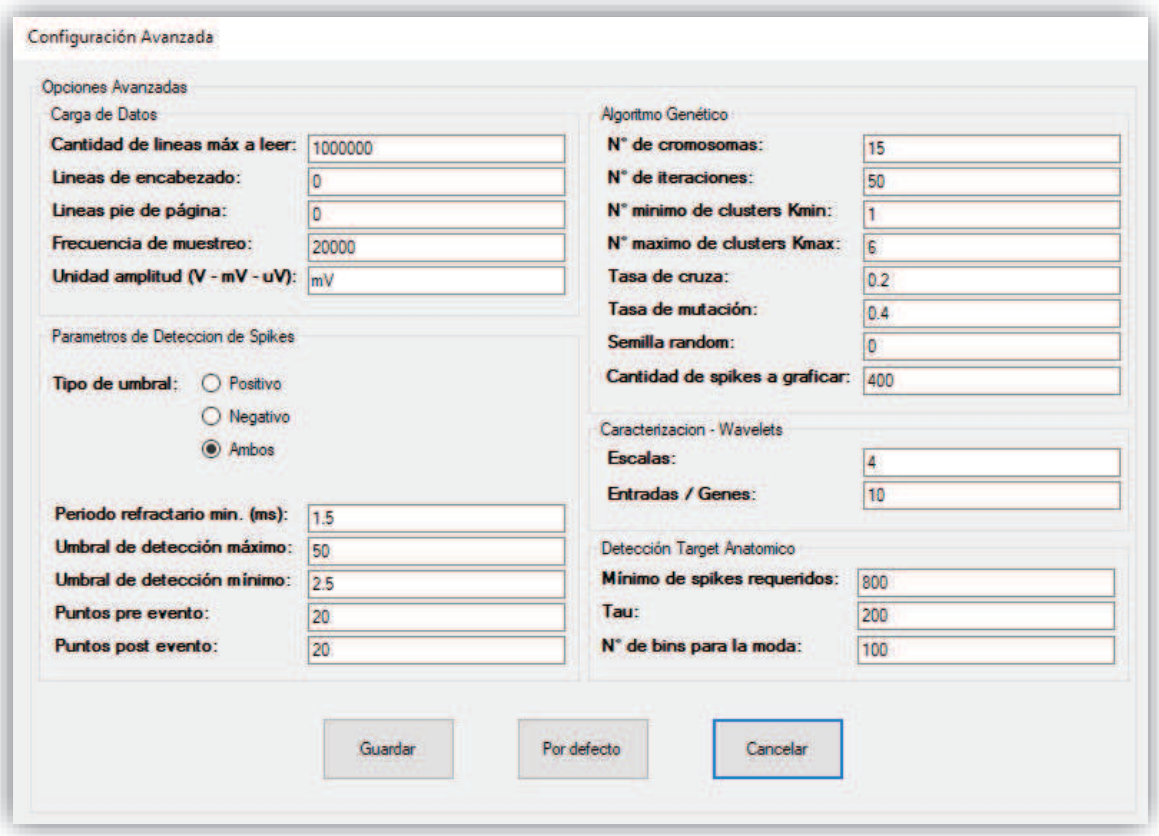


Figura 4: Panel de configuración avanzada. Este panel permite modificar parámetros de todas las etapas del algoritmo. Para carga de datos las opciones son Cantidad de líneas máx a leer, Líneas de encabezado, Líneas de pie de página, Frecuencia de muestreo y Unidad amplitud. Luego para la detección de spikes, el usuario puede definir Tipo de umbral, Periodo refractario, Umbral de detección máximo, Umbral de detección mínimo, Puntos pre evento y Puntos post evento. La utilidad de esto es que se puede adaptar el algoritmo de detección a señales en diversos escenarios. En el caso del algoritmo genético, se puede seleccionar el N° de cromosomas, N° de iteraciones, N° mínimo de clusters Kmin, N° máximo de clusters Kmax, Tasa de cruza, Tasa de mutación, Semilla random y la Cantidad de spikes a graficar. Para el análisis wavelet, el usuario puede modificar las Escalas y las Entradas / Genes. Esto posibilita modificar el nivel de descomposición (escalas) y la cantidad de coeficientes (dimensiones) a calcular en el proceso wavelet. En la sección de detección de target anatómico es posible modificar el Mínimo de spikes requeridos, Tau y N° de bins para la moda. Esta alta flexibilidad en la aplicación permite adaptar el software a posibles y variados escenarios que puedan presentarse como así también extender su rango de uso a tareas para las cuales no fue planificado originalmente. Los valores por defecto se encuentran prefijados y optimizados para la detección de globo pálido durante neurocirugía funcional.

En caso de ser necesario se pueden cargar los valores por defecto y así restaurar los parámetros originales de la aplicación. Esta configuración, es almacenada en un archivo XML que puede ser accedido con un editor de textos para su modificación. La modificación de parámetros requiere que la aplicación sea reiniciada. De lo contrario, la misma utiliza los valores cargados al inicio de la ejecución del programa. La sección de *Configuración Avanzada*, se encuentra subdividida para poder identificar rápidamente el parámetro en cuestión. La flexibilidad de la aplicación en cuanto a rangos de los parámetros utilizados permitirá optimizarla para distintos usos, haciéndola útil para

usuarios con diferente grado de experiencia. La configuración por defecto se encuentra optimizada para la detección de globo pálido durante neurocirugía funcional.

Carga de datos

En la versión actual del software desarrollado sólo es posible procesar archivos con la extensión *.txt* para archivos de texto. Esta extensión engloba a la mayoría de los sistemas de adquisición de datos, ya que actualmente un gran porcentaje permite exportar las capturas a un archivo de texto plano. Al presionar el botón *Cargar Datos* la aplicación consulta por la frecuencia de muestreo y ofrece cambiarla. Acto seguido, se presenta un navegador de directorios del entorno Windows, para poder seleccionar el archivo de origen de datos. Ya seleccionado el archivo, la aplicación comienza a cargar los datos respetando los parámetros que posea configurados en las opciones avanzadas.

Detección de spikes

Luego de la finalización del proceso de carga de datos se habilita el botón *Detectar Spikes.* El mismo permite la búsqueda de potenciales de acción definidos por la superación de dos umbrales (positivos, negativos o ambos) y limitados por una ventana de puntos que se desplaza sobre todos los datos disponibles. Existen dos tipos de umbrales, positivo y negativo, cada uno con un máximo y un mínimo: en total es posible trabajar con 4 umbrales simultáneos. Los umbrales mínimos, positivo y negativo permiten aislar el ruido de base, y los umbrales máximos permiten descartar picos de actividad espurios que no corresponden a la actividad neuronal buscada. Se muestra un ejemplo en la figura 5.

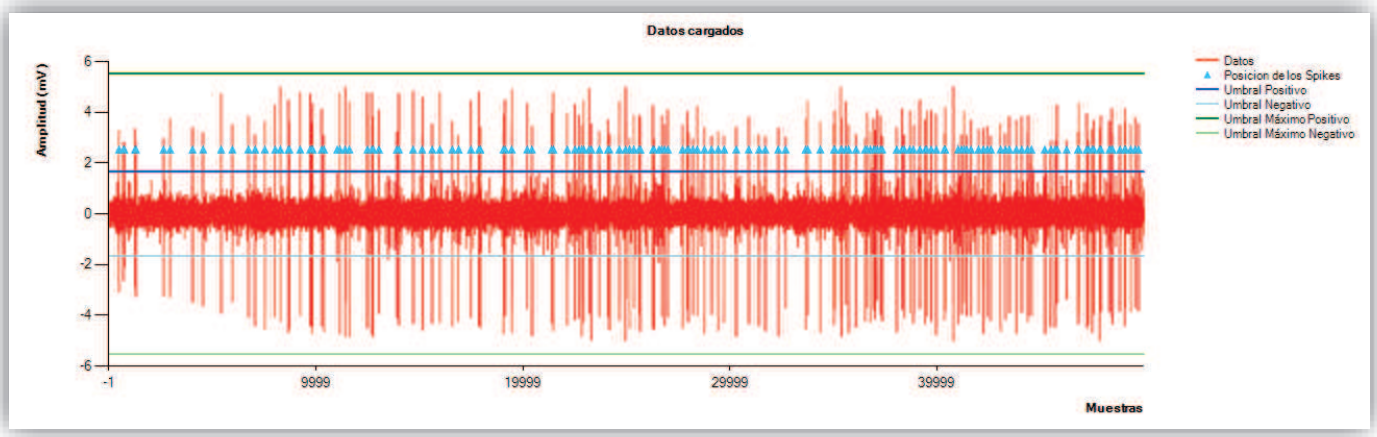


Figura 5: Captura de pantalla tomada del software. Se observa un microregistro de actividad neuronal, con los 4 umbrales definidos permitiendo segmentar la actividad deseada. Los umbrales máximos positivo y negativo (verdes) permiten descartar picos de actividad indeseada; los umbrales mínimos positivo y negativo (azul/celeste) permiten aislar el ruido de base de la señal. Los triángulos celestes sobre la gráfica de la señal (en rojo) indican la posición donde se ha identificado un spike.

La importancia de que el usuario pueda modificar el umbral recae en la posibilidad de remover actividad neuronal de fondo inespecífica como el ruido de base de la señal. Esto se consigue aplicando un primer criterio simple basado en la relación entre la amplitud de los potenciales de acción aceptados en el análisis y el ruido de la señal. Los umbrales están definidos con el cálculo de la desviación estándar del ruido de la señal:

$$Umbral = stdmin * Ruido\_STD \quad (1)$$

dónde el Ruido_STD es calculado como una fracción de

$$Mediana(Abs(señal\_datos)). \quad (2)$$

Un ejemplo de esto se observa en la figura 6. En la misma se observa que, si el umbral se establece superando los 4 mV, quedarían fuera del análisis de detección los 6 spikes claramente observables. Por otro lado, si el umbral es demasiado bajo (inferior a 0.278), podría detectarse el ruido de base como una serie de spikes. Un valor apropiado podría estar comprendido entre 1.086 y 2.701.

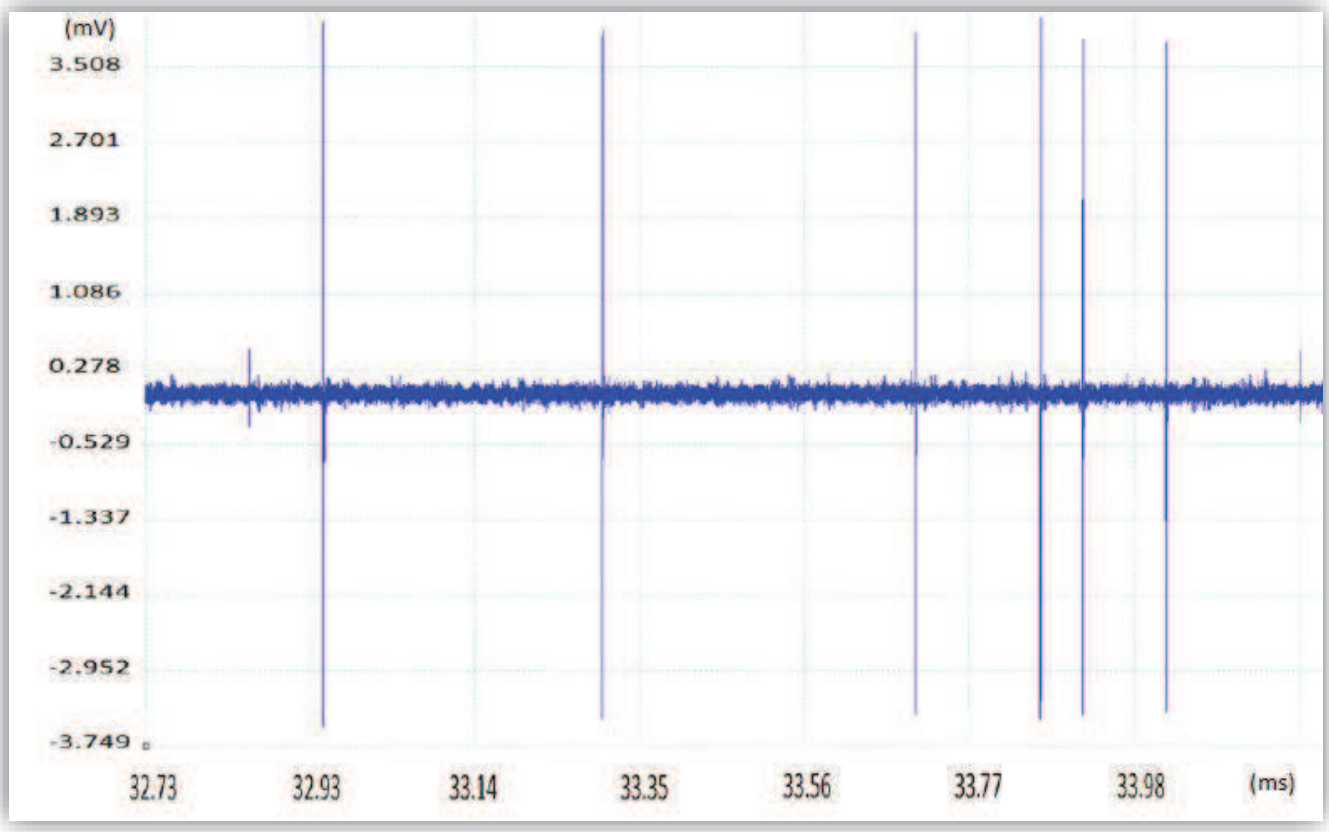


Figura 6: Ejemplo de microregistro neuronal (MER) similar a los utilizados en este proyecto en el set de datos reales. En este ejemplo se puede observar actividad diferenciada de lo que parecen dos neuronas (spike pequeño a los 32.85 y 34.19 ms, spike grande a los 32.94, 33.30, 33.69, 33.79, 33.92 y 34.03 ms) y actividad eléctrica de fondo. La actividad eléctrica corresponde a la actividad promediada de un grupo indeterminado de neuronas que se encuentran a distancias relativamente mayores del electrodo de registro. La actividad de las dos neuronas diferenciadas debe confirmarse como tal realizando spike sorting y clustering.

Los parámetros *tipo de umbrales* (positivo, negativo o ambos), *periodo refractario*, *umbrales de detección máximo y mínimo* y los *puntos pre y post evento* (que conforman la ventana de análisis) son modificables desde el panel de configuración y además se puede variar el valor del *Periodo refractario*, correspondiente al lapso donde la neurona está inactivada para generar otro potencial de acción. Esto permite que el software sea capaz de no considerar spikes que no están separados temporalmente un valor mínimo de un *Periodo refractario* con su predecesor. Los tiempos de ocurrencia de los spikes encontrados pueden ser graficados sobre la señal (si esta fue mostrada), marcando con un triángulo celeste sobre la señal. El software también incluye un recuadro que indica cuántos spikes fueron procesados por el algoritmo.

Análisis wavelet

Una vez detectados los spikes se realiza la caracterización de los mismos a través de la extracción de coeficientes mediante transformada wavelet. Esta transformada está definida como la convolución entre la señal que estamos analizando $x(t)$, y la función wavelet $\psi a, b(t)$:

$$W\psi X(a,b) = \langle x(t) \mid \psi a, b(t) \rangle \quad (3)$$

donde $\psi a, b(t)$ son versiones contraídas y desplazadas de una única función wavelet $\psi(t)$

$$\psi a, b(t) = |a|^{-\frac{1}{2}} \psi\left(\frac{t-b}{a}\right) \quad (4)$$

siendo a y b la escala y la traslación respectivamente (Quiroga, Nadasdy et al. 2004).

Los coeficientes wavelet con los que se modula la función utilizada para la transformada permiten extraer características de los spikes. En nuestro caso el software comienza calculando 10 coeficientes y por lo tanto caracterizando cada spike en un espacio de 10 dimensiones, lo que puede ser modificado desde el panel de C*onfiguración avanzada*. (Ver figura 4, parámetro *Entradas / Genes*). Estas 10 dimensiones luego pasan a ser los 10 genes de cada cromosoma del algoritmo genético (ver más abajo). El algoritmo de análisis wavelet utiliza la función de Haar y comienza con un nivel de descomposición 4. Este nivel puede ser alterado modificando el parámetro *Escalas*, que corresponde al parámetro a de la ecuación (3) y (4). (Ver figura 4, parámetro *Escalas*) (Quiroga, Nadasdy et al. 2004)

Algoritmo genético - Spike sorting

El algoritmo utilizado para realizar la agrupación de spikes en clusters-neuronas está basado en un algoritmo genético. Este algoritmo es un generador de posibles soluciones aleatorias que evolucionan, se cruzan, mutan y se evalúan hasta optimizar el resultado del ordenamiento propuesto (siguiendo los conceptos de la genética natural y los principios de evolución). Los algoritmos genéticos se inspiran en la biología evolutiva que pondera la supervivencia del más apto (Martinez, Servente et al.). Son herramientas de inteligencia artificial que pueden ser utilizadas para agrupar elementos según sus características (genes, en este caso dimensiones procedentes del análisis wavelet). Este tipo de algoritmo posee un indicador o métrica que sirve para evaluar las soluciones propuestas y así utilizarlas o descartarlas. El algoritmo promueve la supervivencia de cada solución según el éxito relativo entre las métricas de las soluciones, aumentando el número de copias de la mejor solución. Posteriormente este algoritmo realiza la cruza y mutación de los cromosomas (soluciones propuestas inicialmente de manera aleatoria) (Maulik and Bandyopadhyay 2000).

El algoritmo desarrollado en este trabajo consiste en los siguientes pasos que se repiten por cada k con $\mathrm{Kmin} <= \mathrm{k} <= \mathrm{Kmax}$:

1. Generación de cromosomas: se proponen soluciones (cromosomas), compuestas de *k* centros para el agrupamiento en clusters del conjunto de datos analizados. Cada cromosoma está compuesto por 10 genes, a los cuales se les asignan k valores (alelos) tomados aleatoriamente de la matriz de características de los spikes. De esta manera, cada cromosoma está formado por k centros propuestos en 10 dimensiones, las cuales se encuentran a escala con la matriz de coeficientes. Esta matriz fue obtenida previamente por el proceso de análisis wavelet (cantidad de genes = cantidad de dimensiones = cantidad de coeficientes). A su vez, por cada iteración (Ver figura 4, *N° de iteraciones*) se realizan los puntos 2 a 6.

2. Métrica: Para cada cromosoma se clasifica a cada spike como perteneciente al clúster a cuyo centro su distancia euclídea es menor. Luego se calcula la métrica de las soluciones propuestas (por cada cromosoma) según la ecuación:

$$M = \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| X_j - Z_i \right\| \quad , (5)$$

dónde Z_i son las coordenadas de cada uno de los centros del cromosoma al que se le está calculando la métrica, X_j es un alelo de uno de los spikes detectados contra el que se está calculando la distancia y C_i es un cluster. Se acepta como mejor solución el cromosoma que posee la métrica más baja de todo el set de cromosomas creado. Utilizar el criterio de minimización de M para definir

el número de clusters de un conjunto de datos desconocidos conlleva el riesgo de caer en overfitting: el algoritmo tiende a generar clusters falsos y a asignarles un spike a cada uno para así hacer tender la métrica a 0. Este es un problema general de los algoritmos genéticos, el que no se encuentra resuelto aún para todas las aplicaciones.

3. El almacenado de la solución de la iteración actual ocurre sólo si ésta posee una métrica mejor (menor) que la retenida hasta el momento y si cumple la condición para ser una solución considerada válida:

$$G(C_{i,}C_n) = \frac{D_i}{D_n} = \frac{\|X_j - Z_i\|}{\|X_j - Z_n\|} \quad (6)$$

para todo $X_j \in C_i$ y donde $G(C_{i,}C_n)$ es una relación entre distancias promedio, siendo D_i la distancia promedio de todos los spikes que pertenecen al clúster *i* hacia los centros de la solución actual y D_n la distancia promedio de todos los spikes que pertenecen al clúster *i,* pero hacia los centros de la nueva solución propuesta. La función G consigue solucionar el problema de overfitting descripto en el punto anterior, ya que exige que las nuevas soluciones propuestas estén a una distancia relativa mínima en todas las dimensiones disponibles (genes) para ser consideradas válidas y así evitar la generación de clusters falsos. Podemos citar la dificultad de resolver el problema de overfitting en (Hruschka, Campello et al. 2009).

4. Generación de copias: sólo si la solución propuesta no fue rechazada según la ecuación (6) se procede a la generación de copias de cada cromosoma propuesto, calculando su relación de éxito con respecto al cromosoma que obtuvo la peor métrica. Por lo tanto, para el *cromosoma j* la relación de copias seria:

$$Relacion\ de\ copias\ j = \frac{M_n}{M_j} \quad (7)$$

donde M_j es la métrica del *cromosoma j* y M_n es la métrica del cromosoma con la peor métrica de todos los propuestos, por ende, la mayor. Con este procedimiento se aumenta la probabilidad de que sobrevivan las mejores soluciones en la próxima iteración del algoritmo. Del nuevo grupo de copia de cromosomas se vuelven a extraer la misma cantidad de cromosomas que el usuario indicó en el panel de configuración (ver figura 4: *N° de cromosomas*).

5. Cruza: con la matriz de cromosomas resultante del paso 4, se realiza la cruza de los genes de los cromosomas, utilizando la tasa de cruza T_c (ver figura 4 *Tasa de cruza)* para decidir aleatoriamente si el cromosoma en cuestión debe cruzarse o no:

$$N° \; aleatorio \leq T_c \; => \; Cruzar \; cromosoma \quad (8)$$

La cruza implica cambiar genes de un cromosoma por los genes de otro, elegidos ambos al azar. La cantidad de genes a intercambiar van desde 1 a la cantidad de *(N° de genes -1).*

6. Mutación: el proceso de mutación recibe la matriz de cromosomas como resultado del proceso de cruza. En este caso, se analiza si cada gen de cada cromosoma debe mutar o no, a través de una tasa de mutación definida por el usuario (ver figura 4 *Tasa de mutación*). La mutación consiste en modificar los alelos del cromosoma, reemplazándolos por valores aleatorios generados dentro del rango de valores que tienen los spikes que están siendo analizados.

$$N° \; aleatorio \leq T_m \; => \; Mutar \; Gen \quad (9)$$

La matriz de cromosomas resultante del proceso de mutación indica la finalización de una iteración del algoritmo. Con esta matriz se ingresará nuevamente a la etapa N° 2 de este algoritmo (Métrica), dando así comienzo a un nuevo ciclo hasta que se alcance el *N° de iteraciones* indicado por el usuario mediante el panel de *Configuración Avanzada.* Recién en ese momento, se incrementará el valor de *k* para volver a comenzar con la etapa N° 1, *Generación de cromosomas*. La cantidad total de iteraciones que realizará la aplicación estará entonces definida por:

$$Cantidad \; total \; de \; iteraciones = \; (Kmax - Kmin) * \; N° \; de \; iteraciones \quad (10)$$

Los valores de Kmin y Kmax, representan la cantidad de clusters mínimo y máximo que se propondrán como solución. Para adquisiciones de datos controladas, donde se puede estimar la cantidad máxima de clusters-neuronas presentes en la señal que está siendo analizada, se podrían utilizar valores de Kmax inferiores a 10. Con esto se evitaría procesamiento de datos posiblemente innecesario. El valor elegido para el *N° iteraciones* tendrá implicancia en la cantidad de veces que se realizará la evolución de la solución propuesta para el valor de k actual. Esto significa que se estará definiendo cuántas veces se va a realizar el proceso de generación de copias, de cruza y de mutación a partir de la mejor solución existente (la solución más apta para subsistir es la que prevalece). El

impacto que pueda generar un número elevado de *cantidad total de iteraciones,* viene determinado por el hardware donde reside la aplicación.

Cabe destacar que la condición impuesta en el punto 3 con la ecuación (6), exige que las nuevas soluciones propuestas estén a una distancia relativa mínima en todas las dimensiones disponibles (genes) para ser consideradas válidas. Distancias más chicas, podrían implicar que se está proponiendo un nuevo clúster de agrupación cuando en realidad los spikes corresponden a un clúster cercano preexistente. Esta condición es una solución original propuesta en este trabajo para solucionar el problema de overfitting. Otro punto en el que el algoritmo desarrollado es novedoso es en la utilización de la dimensión euclídea en 10 dimensiones. Esta es una gran diferencia con la solución implementada en (Quiroga, Nadasdy et al. 2004), la que permite superposición en ciertas dimensiones mientras que haya separación en otras: el algoritmo sólo separa de manera independiente por dimensión. Para realizar las pruebas con los sets de datos artificiales y reales el valor adoptado para la relación *G* de le ecuación (6) fue igual a 0.50. Este valor fue también utilizado para la compilación final del software.

Finalizado el algoritmo genético de spike sorting, se procede a realizar el agrupamiento con la solución final obtenida de todo el proceso anteriormente detallado. Posteriormente, se grafican los 3 primeros clusters que poseen más spikes agrupados en ellos (figura 7). En nuestro caso, un clúster se correspondería con una neurona que fue la generadora de todos los spikes agrupados en él y por eso surge la denominación de clúster-neurona.

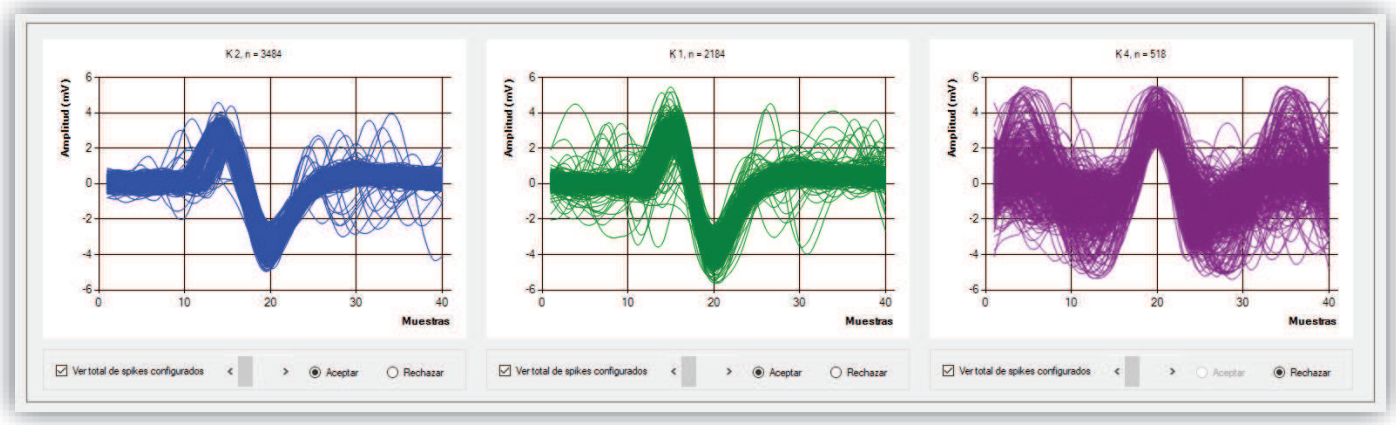


Figura 7: Captura de pantalla del proceso de spike sorting. Una vez finalizado el proceso, el software presenta en tres paneles los 3 clusters (neuronas detectadas, aisladas) con más spikes. Es posible visualizar los potenciales de acción de forma individual utilizando la barra de navegación, o todos juntos seleccionando el checkbox correspondiente que se encuentran ubicados al pie de cada clúster. También se pueden aceptar o rechazar los clusters para que no participen en el siguiente proceso, la detección de target anatómico.

Como se puede observar, el algoritmo genético de spike sorting se apoya fuertemente en la generación de números aleatorios. Para su implementación se utilizaron instancias de la clase *Random* que permite generar números pseudoaleatorios. La misma toma como valor inicial para el algoritmo generador, la cantidad de tiempo en milisegundos que transcurrieron desde el momento en que se inició la computadora (Microsoft Corporation).

Los parámetros que pueden ser modificados por el usuario dentro del algoritmo genético son:

- N° de cromosomas: soluciones propuestas aleatoriamente
- N° de iteraciones: número de veces que se repite el proceso de optimización por cada k propuesto
- N° mínimo y máximo de clusters: define la cantidad de clusters propuestos denominado k, con Kmin <= k <= Kmax)
- Tasa de cruza: varía entre 0 y 1 e indica la probabilidad de que los cromosomas se crucen entre ellos
- Tasa de mutación: varía también entre 0 y 1 pero indica la probabilidad de que un cromosoma mute o no
- semilla random: permite cambiar la semilla con la que se comienza la secuencia de números pseudo-aleatorios generados por la aplicación
- Cantidad de spikes a graficar: determina cuántos spikes se deben dibujar en la gráfica de cada clúster-neurona. (La aplicación sólo muestra como resultado los 3 clusters más poblados)

El proceso de Clustering Genético puede ser re-ejecutado tantas veces como se considere necesario. Debido a que el proceso tiene componentes aleatorios, el resultado del mismo puede variar en cada ejecución.

Datos de validación

Para la validación del algoritmo genético fueron utilizados 2 sets de datos de prueba diferentes: un set de datos artificial generado aleatoriamente y un set de datos perteneciente a pacientes con enfermedad de Parkinson, obtenido durante neurocirugía funcional. Se realizó la validación utilizando 10 dimensiones, valor que puede ser modificado en las opciones avanzadas de configuración bajo la categoría *Entradas/genes*.

El set de datos artificial está compuesto por 3 grupos de puntos generados en MatLAB con la función ***randn***, la que provee números pseudoaleatorios normalmente distribuidos. Los 3 conjuntos de datos poseen un desvío estándar igual a 2 y se diferencian en sus medias con los valores 10, 30 y 50. Por cada valor de la media fueron generados 30 valores, lo que nos brinda un total de 90 puntos. Los mismos son representados a continuación en la figura 8.

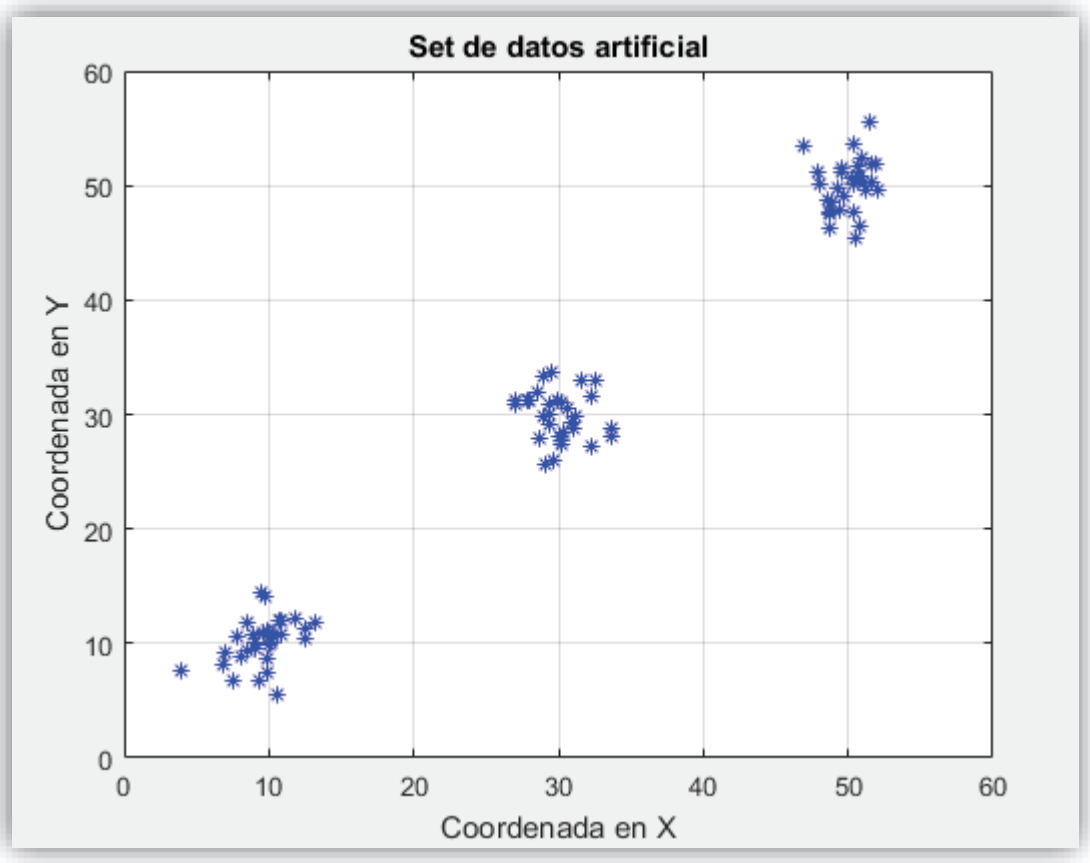


Figura 8: Conjunto de 90 puntos generados pseudoaleatoriamente con desvío estándar = 2 y con media = 10, 30 y 50. Este set de datos artificial fue utilizado para poder validar el algoritmo de clustering genético, debiendo obtener siempre como resultado una solución de agrupación basada en 3 clusters.

El set de datos real está compuesto de 51 microregistros de actividad neuronal de 5 pacientes con enfermedad de Parkinson. Estos registros fueron obtenidos a lo largo de 10 tractos quirúrgicos durante la implantación de electrodos de estimulación cerebral profunda. El método de obtención del microregistro utilizado por el equipo quirúrgico (en nuestro caso del Hospital Fleni) se detalla a continuación brevemente:

- Pacientes despiertos bajo efectos de anestesia local.
- Pacientes sin medicación neurológica en el momento de la cirugía.
- Blanco quirúrgico planificado con MRI y sistema esterotáctico.
- Coordenadas anatómicas corregidas con técnicas de fusión de imágenes MRI.
- Microelectrodos de platino-iridio (80%/20%), impedancia nominal 0.8 - 1.2MΩ.
- Señal Filtrada con filtro pasa banda 300-5000Hz y con un filtro Notch a 50Hz.
- Calibración del sistema con una señal de 1mV al comienzo de cada cirugía.
- Amplificación de la señal, acondicionamiento y digitalización con un sistema de adquisición dedicado. (Fs = 50kHz).

En total se obtuvieron 24 registros de actividad neuronal en el globo pálido externo (GPe) y 27 en el globo pálido interno (GPi). La localización fue determinada por un experto que realiza la evaluación subjetiva del MER basado en su experiencia. Este procedimiento es actualmente el método gold standard para este tipo de intervenciones. Por lo mismo, esta clase de intervenciones están ampliamente supeditadas a las habilidades y experiencia del personal idóneo involucrado. Para la validación del algoritmo con el set de datos reales se trabajó con 100.000 puntos de las señales anteriormente detalladas.

Algoritmo de detección de target anatómico

La detección de target anatómico, corresponde a la implementación del algoritmo desarrollado previamente por la Dra. Andres en colaboración con el Grupo de Movimientos Anormales del Instituto FLENI (MovAn) (Andres, Cerquetti et al. 2017).

Una vez que los spikes han sido agrupados por el algoritmo genético se tienen identificados los clusters-neuronas. Si la cantidad de potenciales de acción supera el número establecido para el parámetro *N° de spikes requeridos* y ese clúster-neurona no es rechazado, se puede proceder a utilizar el botón *Detectar Target.* Este procedimiento, genera una función denominada *función estructura*, a partir de trabajar con los intervalos entre los spikes y los incrementos entre ellos. Se define la función estructura temporal, S_q, como:

$$S_q(\tau) = \langle |\Delta I(\tau)|^q \rangle \quad (11)$$

Para aplicar esta ecuación el primer paso consiste en calcular la diferencia entre cada elemento de la serie y el elemento sucesivo separado por una escala τ. Deben definirse las escalas de interés, típicamente entre 1 y 1000 para estudiar correlaciones de largo plazo. Se obtendrá un numero de diferencias igual al largo de la serie temporal menos la escala (n – τ). El siguiente paso consiste en evaluar el valor absoluto de todas las diferencias obtenidas, elevarlas a la potencia de q y promediarlos (de la misma forma que con τ, debe definirse el rango de interés para q). Este procedimiento debe repetirse para todas las escalas de interés. Una vez obtenida la función estructura, es necesario aplicar un filtro con una ventana móvil de 30 puntos. A partir de esta función suavizada se calcula la primera derivada para obtener S'. En trabajos previos, el comportamiento de S' ha demostrado ser característico de grupos neuronales que es crítico identificar durante neurocirugía (GPe, GPi). En particular, la escala temporal a la cual S' tiende a 0 es característica de determinadas localizaciones anatómicas. El proceso de detección de target busca este punto de inflexión de S' siguiendo el algoritmo publicado en Andres et al. 2015. Por último, los resultados probabilísticos obtenidos se muestran con gráficos de barras 3D y sus respectivas leyendas.

Resultados

Requerimientos de sistema

Basado en el rendimiento de la aplicación y en los resultados obtenidos, se definen a continuación los requerimientos de sistema necesarios para una correcta experiencia de usuario.

Requerimientos mínimos de sistema

- **Procesador:** Intel Core I3 o similar
- **Memoria RAM:** 8 GB
- **Placa de video:** NVIDIA 7600 con 1GB DDR4 o similar
- **Almacenamiento:** Disco rígido SATA 5400rpm o superior
- **Sistema Operativo:** Windows 7/8/10 x86/x64 con .NET Framework 4.5 o superior
- **Pantalla:** Monitor de 17”

Requerimientos recomendados de sistema

- **Procesador:** Intel Core I7 o similar
- **Memoria RAM:** 16 GB
- **Placa de video:** NVIDIA 7600 con 1GB DDR4 o similar
- **Almacenamiento:** Disco SSD (estado sólido)
- **Sistema Operativo:** Windows 7/8/10 x86/x64 con .NET Framework 4.5 o superior
- **Pantalla:** Monitor de 17” o superior

Convergencia del algoritmo genético

El problema que debe resolver el algoritmo genético es el clustering no supervisado de un conjunto de datos compacto en 10 dimensiones, siendo el número de clusters desconocido. Además debe generar y evolucionar una solución con la mínima métrica (ecuación (5)), pero sin producir *overfitting.* Este problema se resuelve aplicando el criterio de distancias relativas (ecuación (6), paso 3 del algoritmo genético). Esta ecuación es muy importante ya que sin ella si *k >= a la dimensión del conjunto de datos*, entonces el conjunto de centros tiende al conjunto de k. Esto significa que se asigna a cada clúster-neurona un spike para hacer tender la *métrica* a *0*. Agregando la condición de distancia relativa mínima (función G, ecuación 6) el algoritmo optimizará una solución y no dependerá del valor de *k*,

desapareciendo así la tendencia al overfitting. Las soluciones con clusters falsos serán consideradas no válidas y descartadas.

Los resultados para *K=10* pueden verse en la figura 9. Este gráfico fue realizado trazando la evolución de la *Métrica vs Iteración* que devolvió el algoritmo, manteniendo todos los parámetros constantes. Los valores asignados fueron 30 cromosomas, N° de iteraciones = 200 (2000 iteraciones totales), tasa de cruza = 0.2, tasa de mutación = 0.4, Kmin =1 y Kmax = 10. Las 2000 iteraciones totales fueron una elección basada simplemente en el costo computacional. Para esta figura se utilizó el set de datos artificial descriptos con anterioridad en la sección ***Datos de validación.*** En rojo se observan las soluciones propuestas y válidas que generó el algoritmo. La curva negra representa las soluciones rechazadas por no cumplir con el criterio de la ecuación (6) para validar soluciones. Por lo tanto, aunque la métrica continúo disminuyendo hasta la iteración 2000 (overfitting), el algoritmo obtuvo la última solución válida cercano a la iteración número 800.

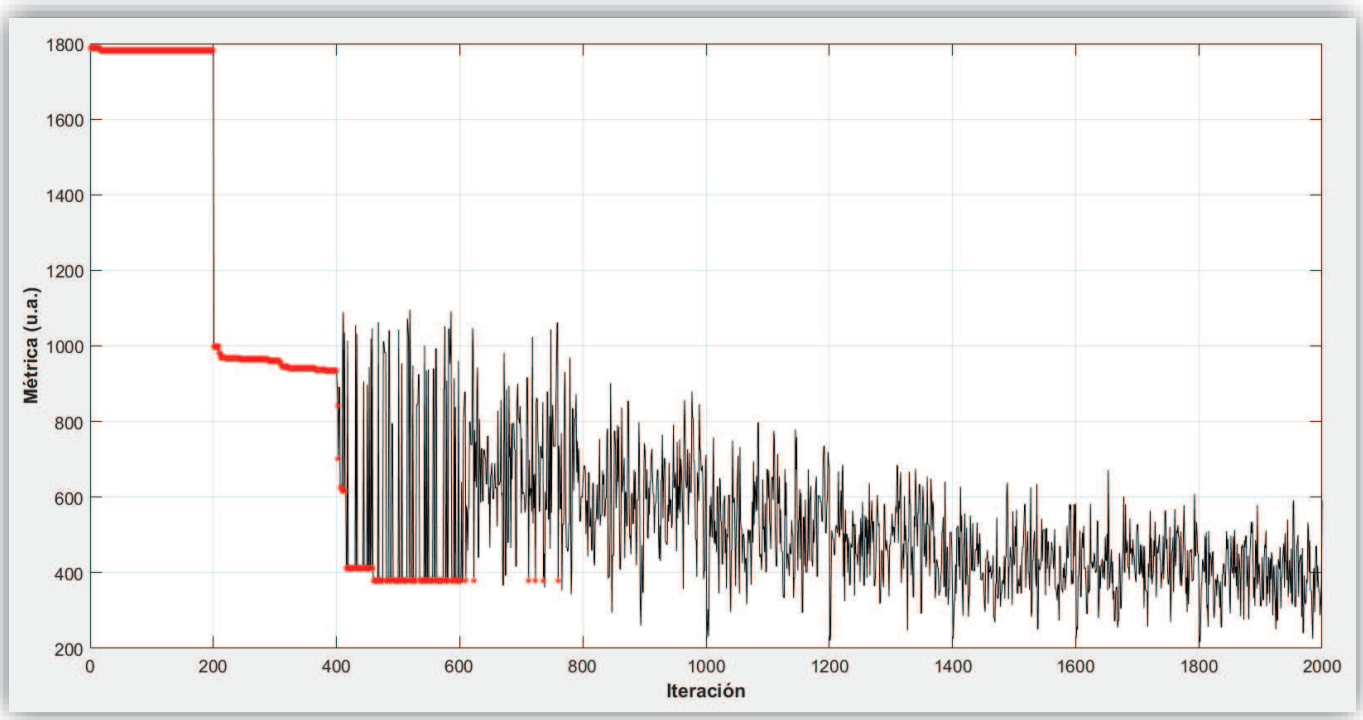


Figura 9: Convergencia del algoritmo genético en la clasificación del set de datos artificiales (3 grupos de datos aleatorios, para más detalle ver texto). En el eje vertical se puede observar la métrica; en el eje horizontal se muestra el número de iteraciones. La línea negra muestra las soluciones consideradas no válidas y por lo tanto rechazadas. Los puntos rojos indican las soluciones válidas que formuló el algoritmo. Como se puede apreciar el algoritmo propuesto ha resuelto el problema de overfitting descartando las soluciones con clusters falsos consideradas no válidas. (30 cromosomas, 2000 iteraciones totales, tasa de cruza = 0.2, tasa de mutación = 0.4 y 1 <= k <= 10.)

En las figuras 10 a 15 se puede apreciar el comportamiento del algoritmo genético utilizando sets de datos reales y artificiales. Se fueron variando distintos parámetros, pero siempre utilizando como base los valores de 30 cromosomas, 2000 iteraciones totales, tasa de cruza = 0.2, tasa de mutación = 0.4 y 1 <= k <= 10.

Se puede advertir en las figuras 10 y 11 que la tasa de cruza no influye en la convergencia del algoritmo con un patrón observable. Aunque la tasa varía entre su valor mínimo (0) y máximo (1) admisible, no hay un patrón que demuestre que a tasas más grandes o más pequeñas el algoritmo converja más rápidamente o no. Tampoco se observa un indicio que indique qué valor de tasa de cruza obtiene la solución óptima en menor número de iteraciones. Lo que se puede destacar es que la tasa de cruza = 0 es la que genera menos oscilaciones en la convergencia del algoritmo como podía suponerse. Por el contrario, con la tasa de cruza = 1 no se puede ratificar que sea la que más perturbaciones le agrega a la convergencia del algoritmo.

Con respecto a la tasa de mutación, ocurre una situación similar que puede ser visualizada en las figuras 12 y 13. Las curvas con las distintas tasas de mutación se cruzan entre ellas a lo largo de las iteraciones sin respetar patrones. No se observa un comportamiento que indique qué valores de tasas podrían beneficiar la convergencia del algoritmo o disminuir el número de iteraciones que se utilizan para encontrar la mejor solución válida. A diferencia de la tasa de cruza, con la tasa de mutación no se puede asegurar que la tasa de valor 0 sea la que genera menos oscilaciones, ya que todas las curvas poseen un comportamiento similar. Tampoco se puede concluir con la tasa de mutación = 1 esperando que sea la que aporta más oscilaciones a la convergencia ya que todas las curvas poseen una estabilidad visiblemente similar.

En las figuras 14 y 15 se puede apreciar el comportamiento del algoritmo genético con respecto a la variación del N° de cromosomas. Se puede decir que hay una leve tendencia a ponderar la cantidad de cromosomas utilizados, ya que a medida que se avanza en las iteraciones del algoritmo se incrementa la diferencia en la mejora de la métrica entre los valores más bajos y los más altos de cromosomas. Esto podría ser un comportamiento esperado ya que a mayor N° de cromosomas el algoritmo propone mayor número de soluciones aleatorias para resolver el problema de spike sorting y, a su vez, cada vez que se incrementa K los cromosomas contienen un centro más para proponer como clúster. Por lo tanto, aunque incrementa el tiempo de procesamiento requerido por el algoritmo genético, es preferible un número más grande de cromosomas para lograr mejorar más la métrica y por ende optimizar más la solución.

Cabe destacar que en todos los casos presentados en las figuras 10 a 15 el algoritmo genético siempre concluyó agrupando los spikes en la misma cantidad de clusters (3 para el set de datos artificiales y 2 para el set de datos reales, que era lo esperado en este caso).

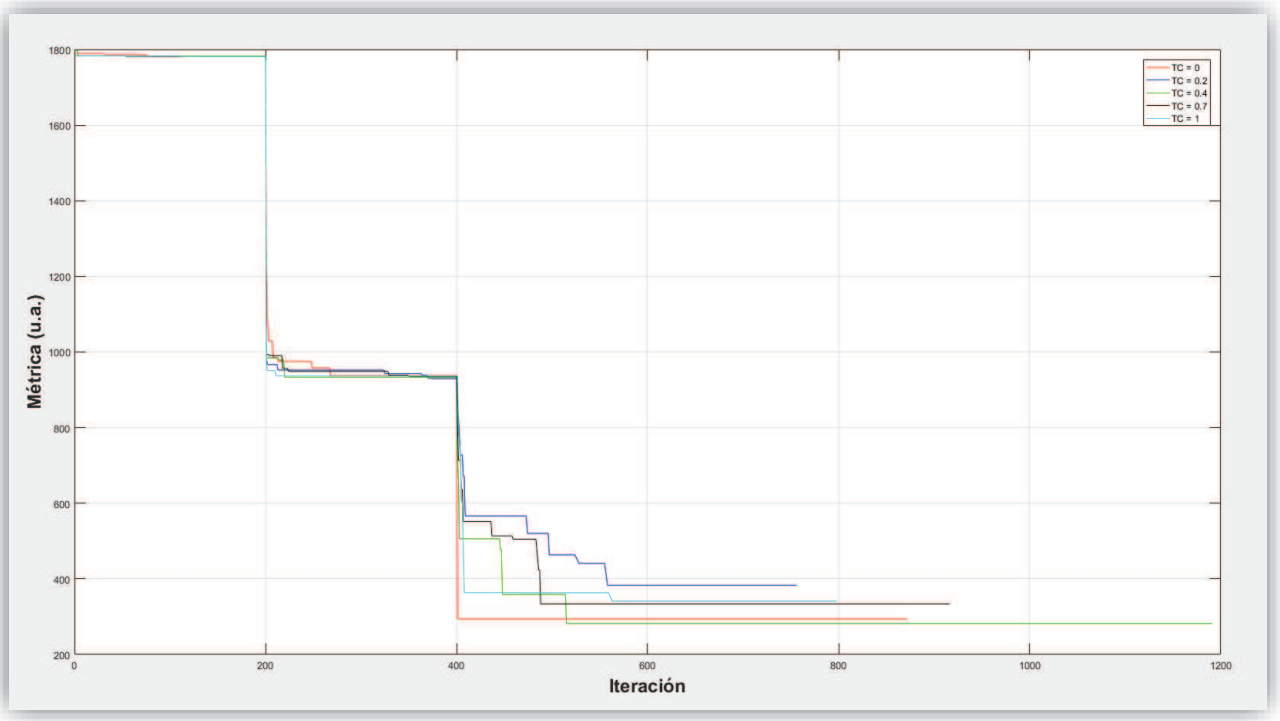


Figura 10: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos artificiales (tres grupos de datos aleatorios, para más detalle ver texto). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones de la tasa de cruza. De este gráfico se desprende que la tasa de cruza no influye en la convergencia del algoritmo con un patrón observable. La tasa de cruza = 0 es la que genera menos oscilaciones en la convergencia del algoritmo como podía suponerse. Por el contrario, con la tasa de cruza = 1 no se puede ratificar que sea la que más perturbaciones le agrega a la convergencia del algoritmo. (30 cromosomas, 2000 iteraciones totales, tasa de mutación = 0.4, 1 <= k <= 10 y variando la tasa de cruza con 0, 0.2, 0.4, 0.7 y 1).

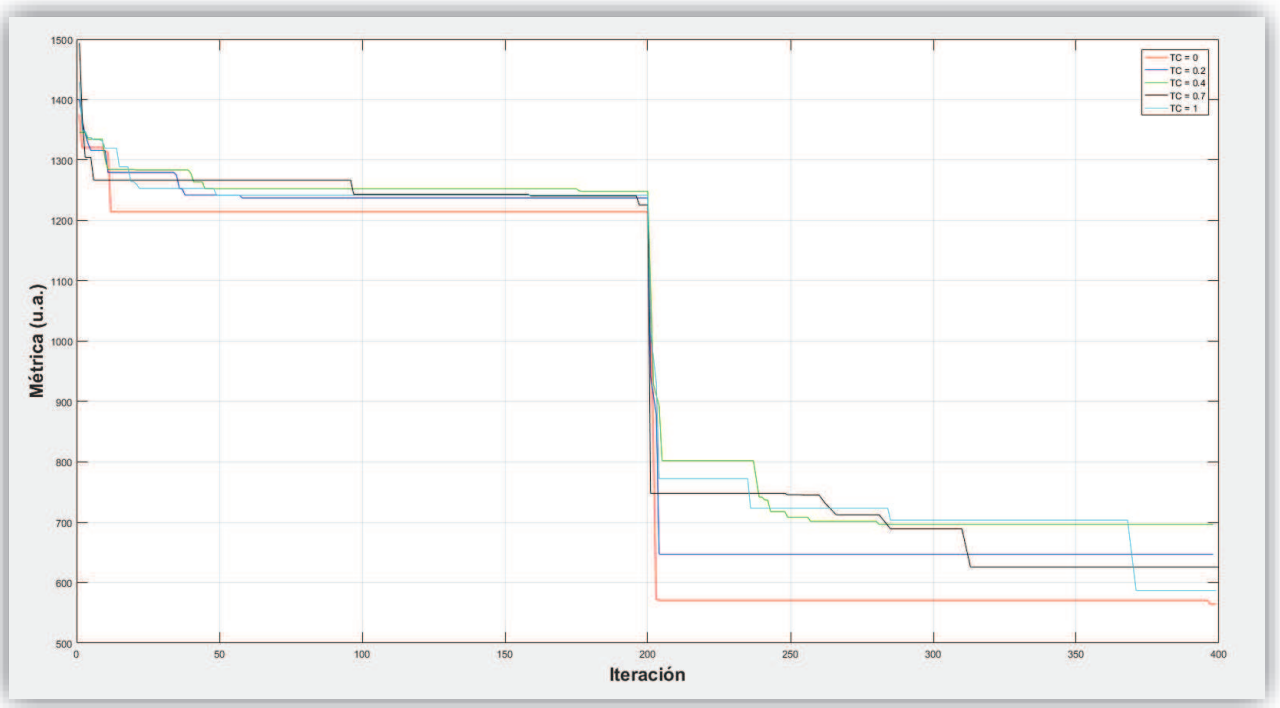


Figura 11: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos reales (para más detalle ver texto, Datos de validación). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones de la tasa de cruza. De este gráfico se desprende que la tasa de cruza no influye en la convergencia del algoritmo con un patrón observable. La tasa de cruza = 0 es la que genera menos oscilaciones en la convergencia del algoritmo como podía suponerse. Por el contrario, con la tasa de cruza = 1 no se puede ratificar que sea la que más perturbaciones le agrega a la convergencia del algoritmo. (30 cromosomas, 2000 iteraciones totales, tasa de mutación = 0.4, 1 <= k <= 10 y variando la tasa de cruza con 0, 0.2, 0.4, 0.7 y 1).

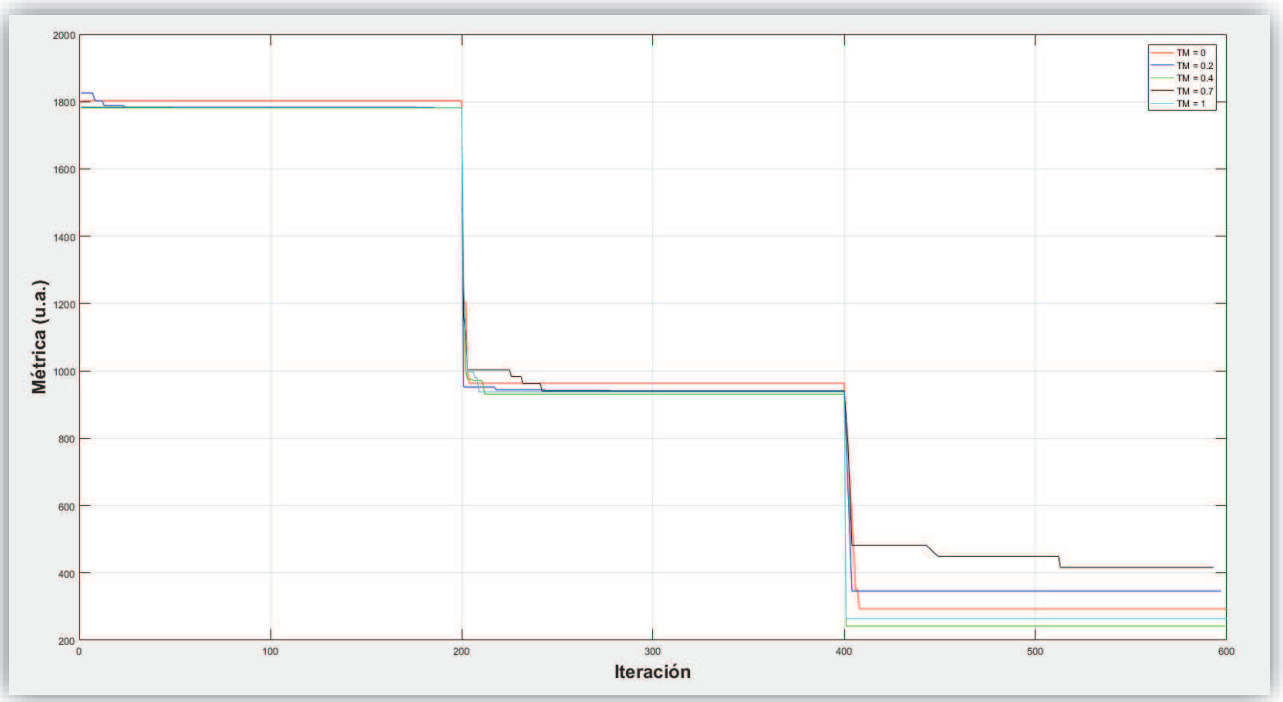


Figura 12: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos artificiales (tres grupos de datos aleatorios, para más detalle ver texto). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones de la tasa de mutación. Las curvas con las distintas tasas de mutación se cruzan entre ellas a lo largo de las iteraciones sin respetar patrones. No se observa un comportamiento que indique qué valores de tasas podrían beneficiar la convergencia del algoritmo o disminuir el número de iteraciones que se utilizan para encontrar la mejor solución válida. Con la tasa de mutación no se puede asegurar que la tasa de valor 0 sea la que genera menos oscilaciones, ya que todas las curvas poseen un comportamiento similar. Tampoco se puede concluir con la tasa de mutación = 1 esperando que sea la que aporta más oscilaciones a la convergencia ya que todas las curvas poseen una estabilidad visiblemente similar. (30 cromosomas, 2000 iteraciones totales, tasa de cruza = 0.2, 1 <= k <= 10 y variando la tasa de mutación con 0, 0.2, 0.4, 0.7 y 1).

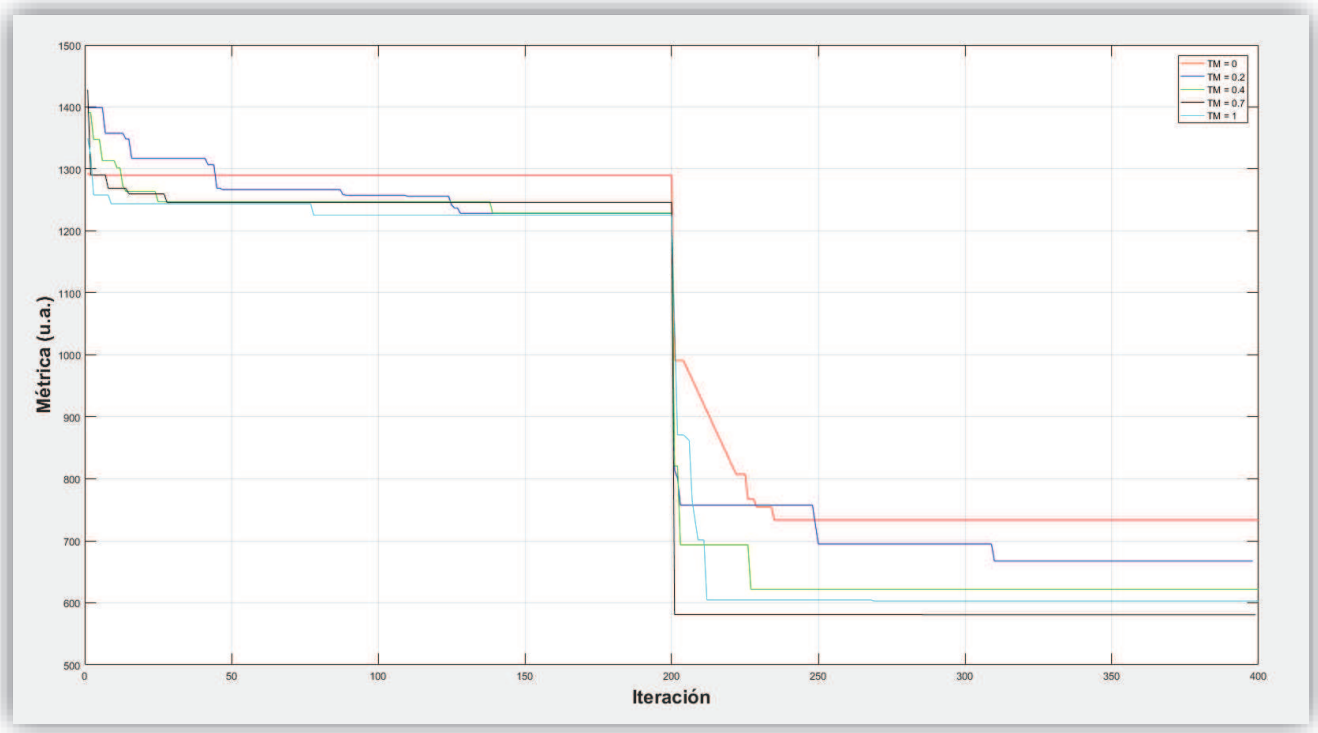


Figura 13: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos reales (para más detalle ver texto, Datos de validación). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones de la tasa de mutación. Las curvas con las distintas tasas de mutación se cruzan entre ellas a lo largo de las iteraciones sin respetar patrones. No se observa un comportamiento que indique qué valores de tasas podrían beneficiar la convergencia del algoritmo o disminuir el número de iteraciones que se utilizan para encontrar la mejor solución válida. Con la tasa de mutación no se puede asegurar que la tasa de valor 0 sea la que genera menos oscilaciones, ya que todas las curvas poseen un comportamiento similar. Tampoco se puede concluir con la tasa de mutación = 1 esperando que sea la que aporta más oscilaciones a la convergencia ya que todas las curvas poseen una estabilidad visiblemente similar. (30 cromosomas, 2000 iteraciones totales, tasa de cruza = 0.2, 1 <= k <= 10 y variando la tasa de mutación con 0, 0.2, 0.4, 0.7 y 1).

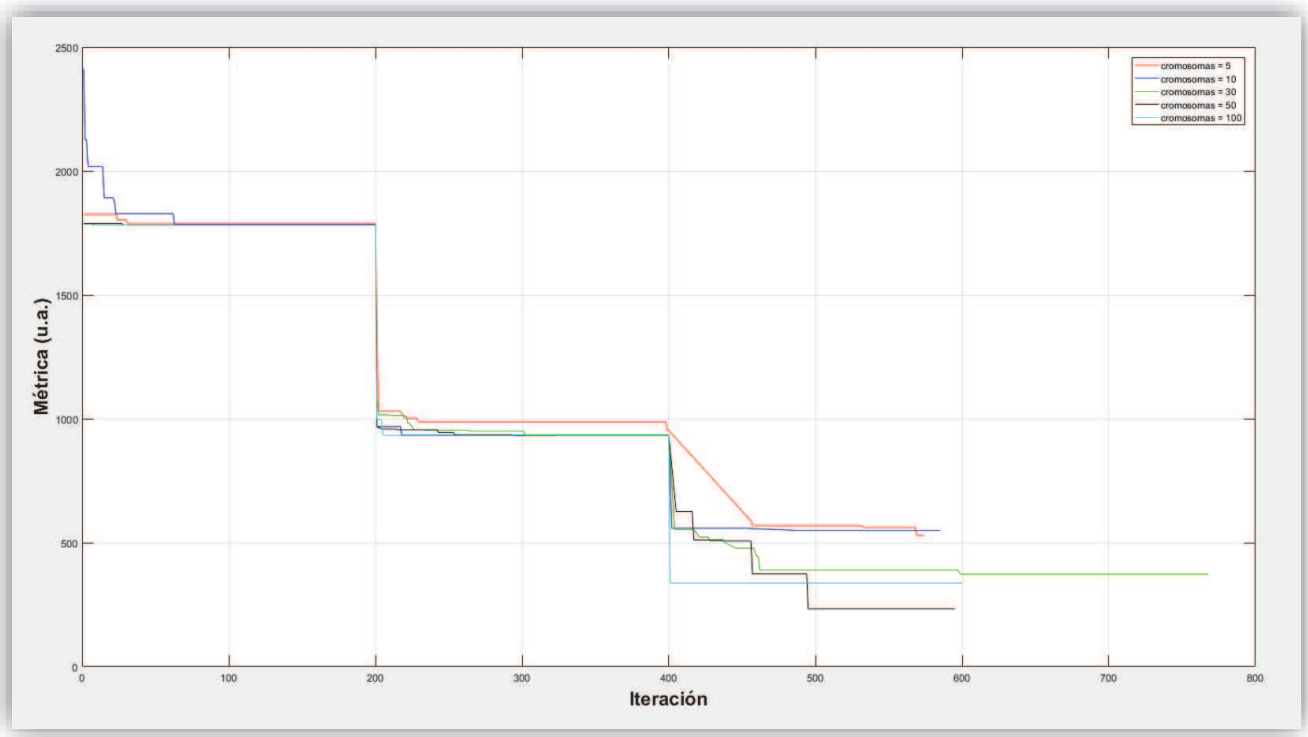


Figura 14: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos artificiales (tres grupos de datos aleatorios, para más detalle ver texto). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones en el N° de cromosomas. Se puede destacar que hay una leve tendencia a ponderar la cantidad de cromosomas utilizados, ya que a medida que se avanza en las iteraciones del algoritmo se incrementa la diferencia en la mejora de la métrica entre los valores más bajos y los más altos de cromosomas. (2000 iteraciones totales, tasa de cruza = 0.2, 1 <= k <= 10, tasa de mutación = 0.4 y variando el N° de cromosomas con 5, 10, 30, 50 y 100).

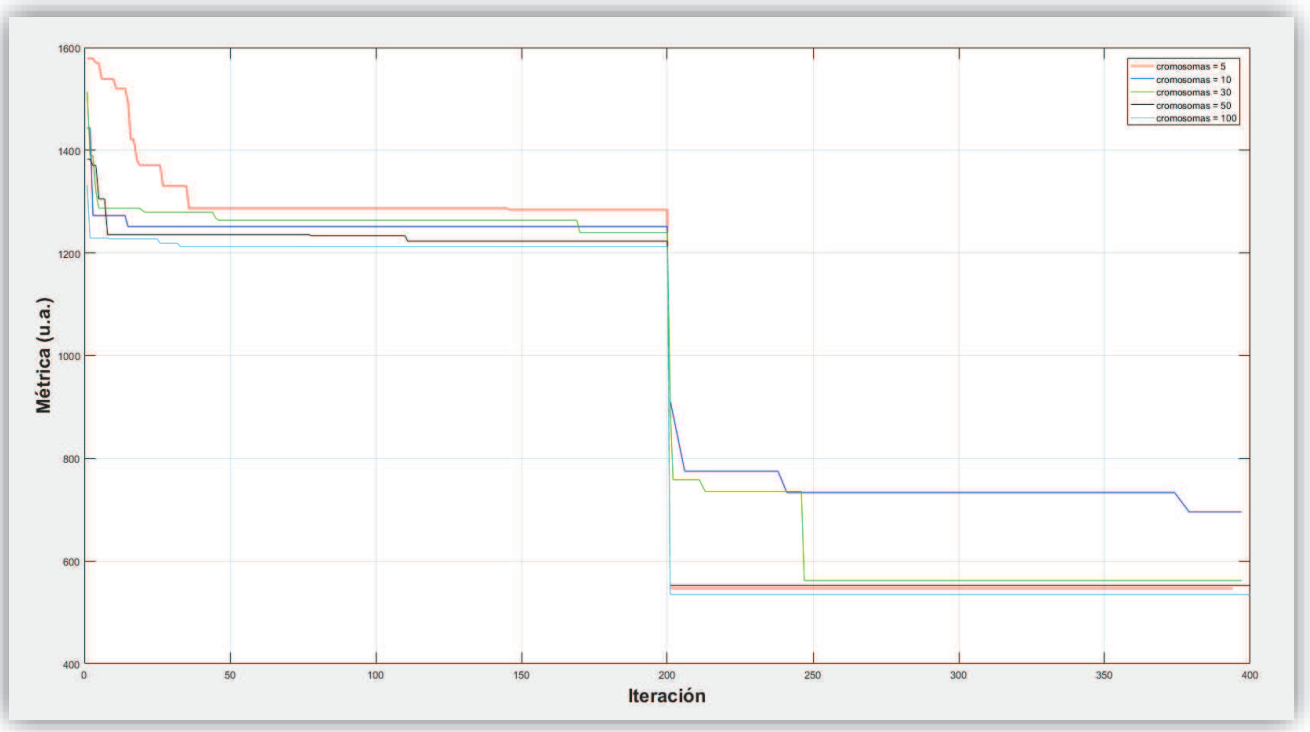


Figura 15: Convergencia de la mejor solución obtenida por el algoritmo genético en la clasificación de datos reales (para más detalle ver texto, Datos de validación). En el eje y se puede observar la métrica; en el eje x se muestra el número de iteraciones. Líneas de distinto color indican variaciones en el N° de cromosomas. Se puede destacar que hay una leve tendencia a ponderar la cantidad de cromosomas utilizados, ya que a medida que se avanza en las iteraciones del algoritmo se incrementa la diferencia en la mejora de la métrica entre los valores más bajos y los más altos de cromosomas. (2000 iteraciones totales, tasa de cruza = 0.2, 1 <= k <= 10, tasa de mutación = 0.4 y variando el N° de cromosomas con 5, 10, 30, 50 y 100).

Para concluir con la sección de resultados se presenta a continuación la figura 16, en la que se muestra un cuadro con las posibles presentaciones de información que puede brindar la aplicación hacia el usuario al finalizar el algoritmo de detección de target anatómico. El cuadro A presenta una probabilidad condicional idéntica para las dos estructuras (GPe/GPi), por lo que sería un caso no concluyente. En la sección B, el algoritmo de detección de target anatómico no detecta un patrón temporal para poder presentar un resultado. En el recuadro C, la posibilidad de que el clúster analizado sea una neurona de estructura GPi no siendo GPe es elevada, lo que indicaría que se podría estar captando el microregistro neuronal muy cerca de esa estructura anatómica. Por último, el cuadrante D muestra el caso en que el usuario ha rechazado el clúster seleccionando *Rechazar* o el mismo no posee el mínimo de spikes necesarios para proceder con el análisis de detección de target anatómico (ver figura 4, *Mínimo de spikes requeridos*).

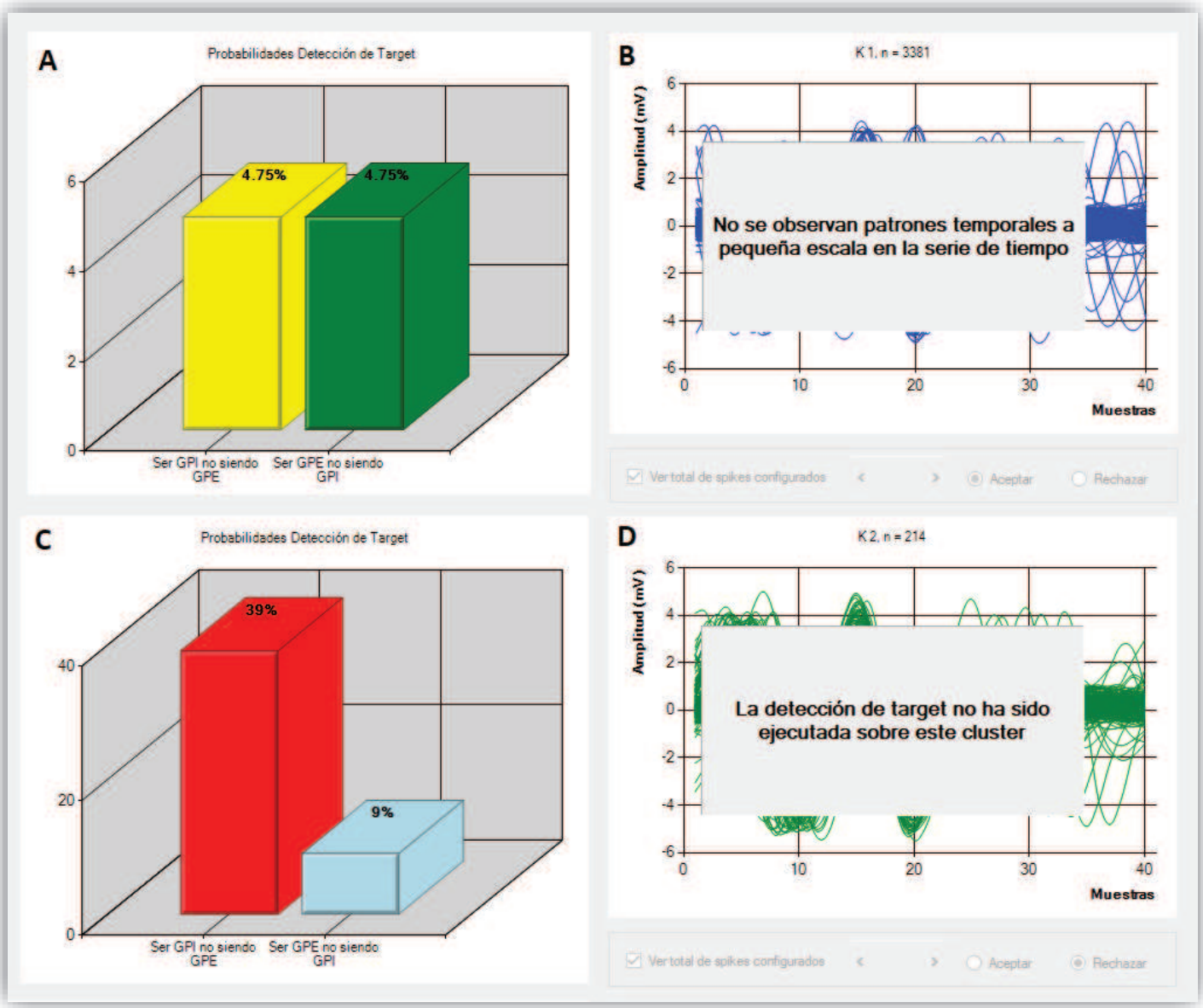


Figura 16: Posibles resultados y presentaciones de información provenientes de la finalización del algoritmo de detección de target anatómico. El cuadro A presenta una probabilidad condicional idéntica para las dos estructuras (GPe/GPi), por lo que sería un caso no concluyente. En la sección B, el algoritmo de detección de target anatómico no detecta un patrón temporal para poder presentar un resultado. En el recuadro C, la posibilidad de que el clúster analizado sea una neurona de estructura GPi no siendo GPe es elevada, lo que indicaría que se podría estar captando el microregistro neuronal muy cerca de esa estructura anatómica. Por último, el cuadrante D muestra el caso en que el usuario ha rechazado el clúster seleccionando Rechazar o el mismo no posee el mínimo de spikes necesarios para proceder con el análisis de detección de target anatómico (ver figura 4, Mínimo de spikes requeridos).

Discusión

El problema de detección automática de blanco anatómico en neurocirugía ha sido abordado a través de los años en numerosas ocasiones y de diferentes maneras (Lozano, Hutchison et al. 1996, Falkenberg, McNames et al. 2006, Cagnan, Dolan et al. 2011). Este problema ha sido resuelto de manera original por la Dra. Andres y equipo en (Andres, Cerquetti et al. 2017), dónde se detalla el algoritmo necesario para poder detectar el blanco quirúrgico a partir de trabajar con los intervalos entre los spikes de las neuronas correspondientes a un mismo tejido. Sin embargo, la implementación clínica de dicho algoritmo requiere un análisis de datos y señales de complejidad superior, ya que es necesario detectar spikes y realizar spike clustering, el cual es la base del algoritmo de Andres et al.

Realizar spike sorting es un problema complejo, ya que se debe agrupar un conjunto de potenciales de acción sin conocer *a priori* la cantidad de clusters a utilizar. Técnicas basadas en algoritmos genéticos han sido propuestas para clustering, sin embargo las mismas tienen el problema general de overfitting (Hruschka, Campello et al. 2009). La determinación de más clusters de los necesarios produce una partición de los clusters existentes en un número pequeño de clusters artificiales. Por otro lado, la detección de menos clusters que los existentes puede provocar la mezcla de spikes correspondientes a distintos tejidos anatómicos, lo que afectaría el desempeño del algoritmo de detección de target. Este proyecto ha resuelto el problema de spike sorting implementando un análisis wavelet de los spikes y un algoritmo genético de clustering en 10 dimensiones (valor que puede ser modificado, ver figura 4, parámetro *Entradas / Genes*). El algoritmo genético resuelve el problema de overfitting con un método original que permite descartar soluciones de agrupamiento según una condición impuesta por la ecuación (6), la que mide distancias relativas entre clusters. Al mismo tiempo el algoritmo continúa optimizando la solución a través del cálculo de una métrica, haciendo prevalecer la mejor y realizando la cruza y la mutación, hasta obtener la mejor solución calculada (ver etapas 4, 5 y 6 del algoritmo genético)(Maulik and Bandyopadhyay 2000). La optimización de la solución entonces dependerá en parte del *N° de iteraciones* configurado (ver figura 4) y a su vez de los valores de los números aleatorios.

El poder de cómputo actualmente disponible en el mercado permite que los tiempos de procesamiento de datos sean cada vez más reducidos. El uso de un hardware superior en capacidad de procesamiento al que fue utilizado para desarrollar este software permitirá incrementar la velocidad de presentación de resultados en tiempo real para brindar apoyo en la toma de decisiones. Otro posible enfoque de este desarrollo podría ser el de realizar una aplicación íntegramente diseñada para procesamiento en paralelo con sistemas distribuidos. Esto permitiría, por ejemplo, aprovechar las salas de informática disponibles en las universidades o laboratorios y así integrar todo el poder individual de cálculo de cada computadora en un solo sistema de procesamiento, mejorando ampliamente la

performance y permitiendo procesar grandes volúmenes de datos. Posteriores modificaciones en la aplicación podrían permitir procesar más formatos de archivos conocidos, como por ejemplo las extensiones de archivos MatLAB (.mat). A su vez, una implementación interesante podría ser la de exportar datos y/o gráficos generados por el software que en este trabajo es presentado. En versiones posteriores de la aplicación sería posible dotar al software de un front-end apto para dispositivos móviles como phablets, handhelds, etc., con menos presentación de datos y enfocado en la optimización del procesamiento debido al bajo poder de cálculo disponible en su hardware. Por otro lado, también sería factible incrementar la cantidad de parámetros configurables y generar perfiles de configuración según el dispositivo o según el operador actual. La posibilidad de modificar aún más parámetros generaría más aceptación en el ámbito académico o de investigación, ya que podría dársele uso o aplicaciones para las cuales no fue desarrollado este software. Características adicionales o funciones extras permitirían a esta aplicación extender su alcance y así poder brindar apoyo en la toma de decisiones sobre diversas patologías.

La integración del software con hardware específicamente diseñado de adquisición de datos podría introducir en el mercado una herramienta de procesamiento de señales que contemplaría todo el camino de la información, desde que es adquirida hasta su procesamiento y posterior presentación de los resultados. Esto permitiría entre otras cosas incrementar el rendimiento, facilitar su utilización y aprendizaje, disminuir la cantidad de equipamiento necesario para el soporte de toma de decisiones y al mismo tiempo abaratar sus costos. De todas formas, debido a que el segmento donde se ubicaría el producto final sería muy específico, debería realizarse previamente un plan de negocios y el estudio de mercado correspondiente.

Las limitaciones de este trabajo están en su mayoría relacionadas con el rendimiento del sistema host de la aplicación. Su velocidad de procesamiento depende ampliamente de las características del soporte físico de donde se extraen los datos. Por tal motivo se recomienda la utilización de un disco de estado sólido (SSD). La memoria RAM disponible en el equipo limita la cantidad de datos que pueden ser procesados de forma simultánea por la aplicación. El incremento en las iteraciones o cantidad de cromosomas utilizados por el algoritmo genético de spike sorting puede afectar sustancialmente el tiempo requerido por el programa para poder presentar un resultado. Como era de suponerse, la aplicación es altamente dependiente de la cantidad de registros a procesar. Actualmente es capaz de procesar más de 150.000 registros en un tiempo inferior a los 60 segundos. Para sets de datos superiores a los 5.000.000 de registros y dependiendo de los parámetros con los que se configura el software podemos esperar tiempos de procesamiento no inferiores a los 20 minutos. Por último, el tamaño del display utilizado, permitirá observar en mayor o menor detalle la información presentada por el software, siendo siempre recomendable utilizar una pantalla de 17” o superior.

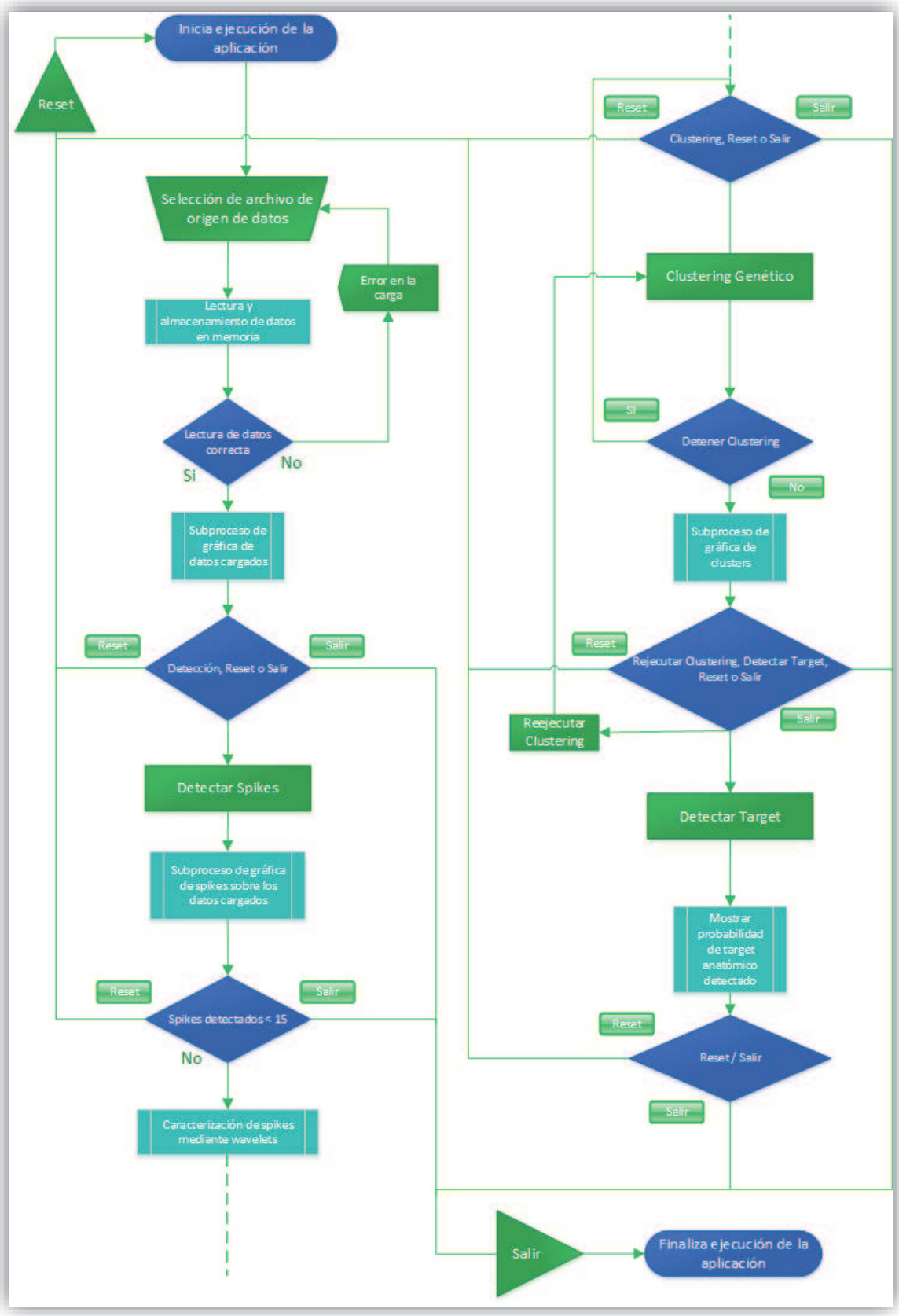
Conclusiones

El software que se desprende de este trabajo final ha alcanzado los objetivos previamente establecidos en el anteproyecto. El mismo permite analizar señales de microregistro neuronal en tiempo real, brindando como resultado una probabilidad utilizada como soporte en la toma de decisiones de la localización del blanco anatómico durante neurocirugía. Dentro de las tareas que puede realizar el software se encuentran la detección, clasificación y agrupación de potenciales de acción (spikes) que luego son procesados con el algoritmo de detección de target anatómico para poder presentar una probabilidad al usuario.

La aplicación cuenta con una interfaz amigable y con un front-end altamente intuitivo para que pueda ser utilizada por usuarios no técnicos. Además, puede funcionar sin conexión a red para ser utilizada en entornos aislados o sin conectividad y pueden ser modificados la mayoría de los parámetros principales que están involucrados en los algoritmos que ejecuta el software. Para finalizar, cabe destacar que el tiempo necesario para obtener un resultado por parte de la aplicación es muy aceptable y se correlaciona ampliamente con el volumen de datos que se desea procesar. Esta última característica permite que el software cumpla con uno de los objetivos principales planteados originalmente, que remarcaba la necesidad de poder utilizarlo como soporte de toma rápida de decisiones durante una intervención neuroquirúrgica.

Apéndice

Diagrama de flujo de la aplicación



Referencias

Anderson, V. C., K. J. Burchiel, P. Hogarth, J. Favre and J. P. Hammerstad (2005). "Pallidal vs subthalamic nucleus deep brain stimulation in Parkinson disease." Archives of neurology **62**(4): 554-560.

Andres, D. S., D. Cerquetti and M. Merello (2011). "Finite dimensional structure of the GPI discharge in patients with Parkinson's disease." Int J Neural Syst **21**(3): 175-186.

Andres, D. S., D. Cerquetti and M. Merello (2017). "Method and apparatus for determining a neuron type. European Patent Office application number EP17160693." Patented invention.

Antroporama.net. "Ganglios de la base." from http://antroporama.net/wp-content/uploads/2013/10/ganglios.gif.

Barbeau, A. (1969). "L-dopa therapy in Parkinson's disease: a critical review of nine years' experience." Canadian Medical Association Journal **101**(13): 59.

Bejjani, B.-P., D. Dormont, B. Pidoux, J. Yelnik, P. Damier, I. Arnulf, A.-M. Bonnet, C. Marsault, Y. Agid and J. Philippon (2000). "Bilateral subthalamic stimulation for Parkinson's disease by using three-dimensional stereotactic magnetic resonance imaging and electrophysiological guidance." Journal of neurosurgery **92**(4): 615-625.

Benabid, A.-L., P. Pollak, A. Louveau, S. Henry and J. De Rougemont (1987). "Combined (thalamotomy and stimulation) stereotactic surgery of the VIM thalamic nucleus for bilateral Parkinson disease." Stereotactic and functional neurosurgery **50**(1-6): 344-346.

Benabid, A. L. (2003). "Deep brain stimulation for Parkinson's disease." Current opinion in neurobiology **13**(6): 696-706.

Björklund, A. and S. B. Dunnett (2007). "Dopamine neuron systems in the brain: an update." Trends in neurosciences **30**(5): 194-202.

Burchiel, K. J., V. C. Anderson, J. Favre and J. P. Hammerstad (1999). "Comparison of pallidal and subthalamic nucleus deep brain stimulation for advanced Parkinson's disease: results of a randomized, blinded pilot study." Neurosurgery **45**(6): 1375-1384.

Cagnan, H., K. Dolan, X. He, M. F. Contarino, R. Schuurman, P. van den Munckhof, W. J. Wadman, L. Bour and H. C. Martens (2011). "Automatic subthalamic nucleus detection from microelectrode recordings based on noise level and neuronal activity." Journal of neural engineering **8**(4): 046006.

Fahn, S. (1996). "Is levodopa toxic?" Neurology **47**(6 Suppl 3): 184S-195S.

Falkenberg, J. H., J. McNames and K. J. Burchiel (2006). "Automatic microelectrode recording analysis and visualization of the globus pallidus interna and stereotactic trajectory." Stereotactic and functional neurosurgery **84**(1): 28-34.

Falkenberg, J. H., J. McNames, J. Favre and K. J. Burchiel (2006). "Automatic analysis and visualization of microelectrode recording trajectories to the subthalamic nucleus: preliminary results." Stereotactic and Functional Neurosurgery **84**(1): 35-45.

Frades-Payo, B., M. Forjaz and P. Martínez-Martín (2009). "Situación actual del conocimiento sobre calidad de vida en la enfermedad de Parkinson: I. Instrumentos, estudios comparativos y tratamientos." Rev Neurol **49**: 594-559.

Goetz, C. G. (2011). "The history of Parkinson's disease: early clinical descriptions and neurological therapies." Cold Spring Harbor perspectives in medicine **1**(1): a008862.

Guridi, J., A. Gorospe, E. Ramos, G. Linazasoro, M. C. Rodriguez and J. A. Obeso (1999). "Stereotactic targeting of the globus pallidus internus in Parkinson's disease: imaging versus electrophysiological mapping." Neurosurgery **45**(2): 278-289.

Guridi, J., M. Rodriguez-Oroz, A. Lozano, E. Moro and A. Albanese (2000). "Targeting the basal ganglia for deep." Neurology **55**(6): S21-S28.

Guridi, J., M. C. Rodriguez-Oroz and M. Manrique (2004). "Tratamiento quirúrgico de la enfermedad de Parkinson." Neurocirugía **15**(1): 5-16.

Hornykiewicz, O. (2006). "The discovery of dopamine deficiency in the parkinsonian brain." Journal of Neural Transmission Supplementum **70**: 9.

Hruschka, E. R., R. J. G. B. Campello, A. A. Freitas and A. C. P. L. F. d. Carvalho (2009). "A Survey of Evolutionary Algorithms for Clustering." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **39**(2): 133-155.

Jankovic, J. (2008). "Parkinson's disease: clinical features and diagnosis." Journal of Neurology, Neurosurgery & Psychiatry **79**(4): 368-376.

Juri, C. and P. Chaná (2006). "Levodopa en la enfermedad de Parkinson:¿ Qué hemos aprendido?" Revista médica de Chile **134**(7): 893-901.

Kumar, R., A. Lozano, E. Montgomery and A. E. Lang (1998). "Pallidotomy and deep brain stimulation of the pallidum and subthalamic nucleus in advanced Parkinson's disease." Movement disorders: official journal of the Movement Disorder Society **13**: 73-82.

Letelier, J. C. and P. P. Weber (2000). "Spike sorting based on discrete wavelet transform coefficients." Journal of neuroscience methods **101**(2): 93-106.

Lewicki, M. S. (1998). "A review of methods for spike sorting: the detection and classification of neural action potentials." Network: Computation in Neural Systems **9**(4): R53-R78.

Lozano, A., W. Hutchison, Z. Kiss, R. Tasker, K. Davis and J. Dostrovsky (1996). "Methods for microelectrode-guided posteroventral pallidotomy." Journal of neurosurgery **84**(2): 194-202.

Machado, A., A. R. Rezai, B. H. Kopell, R. E. Gross, A. D. Sharan and A. L. Benabid (2006). "Deep brain stimulation for Parkinson's disease: surgical technique and perioperative management." Movement disorders **21**(S14).

Martinez-Martin, P. and G. Deuschl (2007). "Effect of medical and surgical interventions on health-related quality of life in Parkinson's disease." Movement disorders **22**(6): 757-765.

Martinez, R. G., M. Servente and D. Pasquini Sistemas Inteligentes.

Maulik, U. and S. Bandyopadhyay (2000). "Genetic algorithm-based clustering technique." Pattern recognition **33**(9): 1455-1465.

Meyers, R. (1942). "The modification of alternating tremors, rigidity and festination by surgery of the basal ganglia." Res Publ Assoc Res Nerv Ment Dis **21**(6): 02-665.

Microsoft Corporation. "A random number generator." from http://referencesource.microsoft.com/#mscorlib/system/random.cs,bb77e610694e64ca.

Obeso, J. A., M. C. Rodriguez-Oroz, P. Chana, G. Lera, M. Rodriguez and C. Olanow (1999). "The evolution and origin of motor complications in Parkinson's disease." Neurology **55**(11 Suppl 4): S13-20; discussion S21-13.

Parkinson's Disease Foundation. (2017). from http://www.pdf.org/parkinson_statistics.

Parkinson's Disease Foundation. (2017). from http://www.pdf.org/about_pd.

Quiroga, R. Q., Z. Nadasdy and Y. Ben-Shaul (2004). "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering." Neural computation **16**(8): 1661-1687.

Rodriguez-Oroz, M., J. Obeso, A. Lang, J.-L. Houeto, P. Pollak, S. Rehncrona, J. Kulisevsky, A. Albanese, J. Volkmann and M. Hariz (2005). "Bilateral deep brain stimulation in Parkinson's disease: a multicentre study with 4 years follow-up." Brain **128**(10): 2240-2249.

Sandoval, L., F. Jiménez, J. Soto, F. Velasco, J. Carrillo-Ruiz, P. Gómez and R. Suárez (2010). "Resultados del tratamiento quirúrgico de la enfermedad de Parkinson en la Unidad de Neurocirugía Funcional, Estereotaxia y Radiocirugía, del Hospital General de México en el período de 1992 a 2009." Rev Mex Neuroci **11**(1): 20-25.

Schrag, A., M. Jahanshahi and N. Quinn (2000). "How does Parkinson's disease affect quality of life? A comparison with quality of life in the general population." Movement Disorders **15**(6): 1112-1118.

Schuurman, P. R., D. A. Bosch, P. M. Bossuyt, G. J. Bonsel, E. J. Van Someren, R. M. De Bie, M. P. Merkus and J. D. Speelman (2000). "A comparison of continuous thalamic stimulation and thalamotomy for suppression of severe tremor." New England Journal of Medicine **342**(7): 461-468.

Shoham, S., M. R. Fellows and R. A. Normann (2003). "Robust, automatic spike sorting using mixtures of multivariate t-distributions." Journal of neuroscience methods **127**(2): 111-122.

Trépanier, L. L., R. Kumar, A. M. Lozano, A. E. Lang and J. A. Saint-Cyr (2000). "Neuropsychological outcome of GPi pallidotomy and GPi or STN deep brain stimulation in Parkinson's disease." Brain and cognition **42**(3): 324-347.