Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería
Maestría y Doctorado en Ciencias e Ingeniería

Algoritmo inteligente para la optimización del proceso de carga en una batería LiPo

Tesis para obtener el grado de Maestro en Ciencias

Presenta:

Osbel Alejandro Islas Silvas

Bajo la dirección:

Dr. Mauricio Alonso Sánchez Herrera

Co-dirigido por:

Dr. Leocundo Aguilar Noriega

27 de octubre de 2019

Agradecimientos

A mis padres Rosa Isela Islas Silvas y Julian Fajer Gonzalez que siempre me apoyaron incondicionalmente ante cualquier decisión que yo tomara, su apoyo me hizo seguir adelante y nunca darme por vencido, a mi hermana Roxana Ivonne Islas Silvas que con su alegría y apoyo me hacía seguir en el mismo camino para ser un buen ejemplo digno de su respeto y admiración, también agradecerle porque me dio uno de mis principales motores en la vida, mi sobrina Ivanna Guadalupe Zavala Islas a la que amo con todo mi corazón. A mi abuelita que siempre se preocupaba por mi salud y bienestar emocional Sonia Lucia Gonzalez Loaiza. A mi novia Gloria Lorena Camacho Lopez que siempre estuvo conmigo en los momentos más difíciles siempre buscando la forma para ayudarme a salir de ellas, inspirándome a ser una mejor persona y formar parte de mis metas y objetivos. A mi director de tesis Dr. Mauricio Alonso Sánchez Herrera que siempre buscaba ayudarme con una sonrisa en el rostro y enseñándome una de las principales lecciones de vida con las que camino de ahora en adelante, trabajar siempre en lo que te hace feliz con perseverancia, respeto, responsabilidad y siempre buscando el bien común, ayudando a las personas que lo necesitan. A mi codirector de tesis Dr. Leocundo Aguilar Noriega el cual considero un ejemplo a seguir desde hace varios años por sus logros y su vasto conocimiento en temas de ingeniería. A mis profesores y sinodales que apoyaron a mi crecimiento intelectual durante toda la maestría y agradeciendo siempre su disposición para ayudarme ante cualquier duda o complicación. A mis compañeros de laboratorio con los que compartí un sin número de buenos momentos, apoyándonos siempre en nuestras deficiencias convirtiéndolas en fortalezas. Pero en especial quisiera agradecer a mi abuelita Maria Bertila Silva Lopez, que, aunque ya no se encuentra aquí con nosotros, siempre está presente en todas mis decisiones, objetivos, metas y logros, su mayor regalo fue enseñarme la importancia que se le debe dar a las personas que amas, independientemente de las cosas materiales o los problemas por los que se pueda atravesar, siempre debemos estar para nuestros seres queridos, porque no sabemos cuándo podrían irse sin haber disfrutado al máximo su presencia.

1. Resumen

Desde que los sistemas electrónicos de uso diario comenzaron a utilizar baterías para funcionar sin la necesidad de estar conectados a una fuente estática de energía, surgió la necesidad de buscar formas para prolongar la duración de estas mismas y así obtener una mayor eficiencia del dispositivo. Actualmente contamos con muchos tipos de baterías que ofrecen diversos beneficios en cuestión del uso de la batería, uno de ellos es que estas baterías son recargables. Las baterías recargables llegaron a solucionar el problema de estar cambiando las baterías cada vez que estas se quedaran sin energía, sin embargo, el proceso de carga de una batería se convirtió rápidamente en un tema de interés para todos aquellos que quisieran maximizar la eficiencia de un dispositivo electrónico. Existen muchos métodos de carga en la actualidad que permiten cargar una batería en un lapso de tiempo bastante corto, en esta tesis, se busca implementar un nuevo método de carga basado en los ya existentes con la ayuda de un algoritmo de inteligencia artificial que permita un mayor entendimiento del comportamiento de las baterías de tipo LiPo y minimizar su tiempo de carga en comparación con los métodos de carga actuales.

Universidad Autónoma de Baja California
FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

FOLIO No. 285

Tijuana, B. C., a 02 de Octubre 2019

C. Osbel Alejandro Islas Silvas
Pasante de: Maestro en Ciencias
Presente

POR LA REALIZACIÓN PLENA DEL HOMBRE

El tema de trabajo y/o tesis para su examen profesional, en la Opción TESIS
Es propuesto, por los C.C. Dr. Mauricio Alonso Sánchez Herrera **y** Dr. Leocundo Aguilar Noriega
Quienes serán los responsables de la calidad de trabajo que usted presente, referido al tema "Algoritmo inteligente para la optimización del proceso de carga en una batería LiPo"
El cual deberá usted desarrollar, de acuerdo con el siguiente orden:

I.- RESUMEN
II.- ÍNDICE DE FIGURAS
III.- ÍNDICE DE TABLAS
IV.- INTRODUCCIÓN
V.- MARCO TEÓRICO
VI.- DESARROLLO
VII.- RESULTADOS
VIII.- CONCLUSIÓN
IX.- REFERENCIAS

Dr. Mauricio Alonso Sánchez Herrera
Director de Tesis

Dr. Leocundo Aguilar Noriega
Co-Director de Tesis

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE CIENCIAS QUIMICAS E INGENIERIA
COORDINACION DE POSGRADO E INVESTIGACION

Dr. José Luis González Vázquez
Director

Dra. Rocío Alejandra Chávez Santoscoy
Sub-Directora

Índice

2. Índice de figuras

3. Índice de tablas

4. Introducción

4.1. Problemática

El uso de dispositivos digitales crece de manera exponencial con respecto al tiempo[Coughlin, 2017], para que estos dispositivos tengan la autonomía de funcionar sin la necesidad de estar conectados a la corriente eléctrica se utilizan baterías. La importancia de estos dispositivos en la vida cotidiana se a vuelto una necesidad, según datos del INEGI el numero de personas que usan un smartphone hasta el 2017 fue de 64.7 millones [INEGI and ENDUTIH, 2018], estos dispositivos funcionan a base de baterías recargables por lo que es necesario someterlos a un proceso de carga de batería. Otro ejemplo es el famoso termino IoT (Internet de las Cosas; del inglés Ïnternet of Things") el cual se define como la interacción y comunicación de los diferentes objetos que nos rodean para alcanzar un determinado objetivo[Atzori et al., 2010], existe una tendencia hacia el uso de baterías recargables para estos dispositivos, así permitiendo, la autonomía necesaria. La principal necesidad que podemos detectar en el uso de estos dispositivos, es la de tener que recargar las baterías y así permitir que pueda seguir operando en la forma correcta, el proceso de cargar una batería toma en la mayoría de los cargadores convencionales bastante tiempo y el tiempo aumenta si la batería tiene una mayor capacidad de carga. En la actualidad podemos encontrar dos tipos de de cargadores, los cargadores convencionales que funcionan en base a un método de carga tradicional y los cargadores de carga rápida, estos últimos implementan métodos distintos al convencional y permiten cargar más rápido y de forma más eficiente una batería. El tiempo de diferencia entre estos dos cargadores es bastante grande e importante, ya que el tiempo de funcionamiento del dispositivo aumenta gracias al menor tiempo que este debe ser sometido al proceso de carga de la batería. El objetivo principal de esta tesis es reducir el tiempo de carga de la batería para aumentar el tiempo de actividad o productividad de un dispositivo, proponiendo una solución eficiente basada en una técnica de inteligencia computacional como algoritmos genéticos para la optimización del proceso de carga de una batería.

4.2. Hipótesis

Mediante el uso de técnicas de optimización basadas en algoritmos genéticos, es posible reducir el tiempo de carga de una batería de tipo LiPo, determinando los parámetros de carga como corriente y tiempo, aplicándolos de forma sistemática en un método de carga.

4.3. Objetivo General

Aumentar el tiempo de actividad o productividad de los dispositivo digitales que funcionan con baterías de tipo LiPo.

4.4. Objetivo Específico

Reducir el tiempo de carga de una batería de tipo LiPo aplicando técnicas de optimización basadas en algoritmos genéticos.

4.5. Metas

1. Crear un algoritmo genético que obtenga los parámetros óptimos para reducir el tiempo del proceso de carga.

2. Crear un prototipo físico(Hardware) para el control de las baterías durante el proceso de carga.

3. Evaluar un método de carga con parámetros generados mediante el algoritmo genético y analizando su eficiencia en comparación con el método de carga convencional.

4. Con los resultados obtenidos, hacer un análisis estadístico de las variables involucradas y consideración de nuevas variables.

5. Marco Teórico

El proceso de carga de una batería va más allá de solamente administrarle energía o corriente, existe una serie de pasos o subprocesos que complementan esta acción. Esta serie de pasos o subprocesos pueden ser modificados de tal forma que cada cambio realizado se ve reflejado directamente en la eficiencia del proceso de carga. La eficiencia puede medirse con el tiempo de carga que conlleva cargar una batería pasando de estar completamente descargada a completamente cargada, el cual tiene una mayor eficiencia si el tiempo de carga fue menor. El tiempo de carga es la consecuencia de la forma en la que se suministra energía a la batería, a esto se le llama método de carga (cita). El método de carga es un procedimiento que debe contemplar un balance entre la seguridad de la batería, los ciclos de vida y el tiempo de carga[Liu et al., 2016].

5.1. Baterías

Los dispositivos que funcionan a base de baterías se han convertido en elementos necesarios para la vida moderna. Esto incluye por ejemplo dispositivos de comunicación, navegación, sensores de control remotos y dispositivos multimedia. En el caso de estos dispositivos, es más práctico y económico el uso de baterías recargables en vez de estar cambiando baterías cada vez que se descargan.

5.1.1. Definición

Una batería eléctrica es un dispositivo que se compone de una o más celdas electroquímicas con conexiones externas, el cual permiten proveer electricidad. Estas celdas galvánicas o voltaicas consisten en dos electrodos inmersos en un material conductor tal como un liquido electrolítico o puente salino. Cuando los dos electrodos son conectados mediante un cable, la corriente comienza a fluir de uno al otro, produciendo lo que conocemos como corriente eléctrica[Crompton and TR, 2000].

5.1.2. Operación básica de una batería recargable

Una batería convierte la energía química en energía eléctrica mediante un proceso de reducción-oxidación (redox) entre los elementos activos. Estos elemento son el agente oxidante, agente reductor y el electrolito. Durante la de descarga, la batería es capaz de entregar energía gracias al proceso redox que sufren los electrodos y cuando se somete a la carga de la batería, el proceso funciona de forma inversa[Tar and Fayed, 2016].

5.1.2.1. Proceso de Reducción y Oxidación (Redox)

Durante el proceso de reducción y oxidación la energía química se transforma en energía eléctrica permitiendo a otros dispositivos aprovecharla para su funcionamiento. Como se menciona anteriormente, existen 3 principales actores en este proceso, el cátodo y el ánodo que funcionan como las terminales positivas y negativas respectivamente, mientras el electrolito funciona como un medio neutro permitiendo que suceda el proceso redox como se muestra en la Fig. 1. Entre los dos electrodos existe una diferencia de potencial lo que permite que corriente fluya de un lado a otro, en condiciones de circuito abierto, la diferencia de potencial de la batería es igual a la suma algebraica de las dos diferencias de potencial de los elementos químicos o electrodos[Hill, Jonh, 1999]. Durante el proceso de descarga de la batería, el ánodo funge como agente reductor, este se encarga de entregar electrones (Fig. 1(a)) por lo que sufre una oxidación, por el caso contrario el cátodo funge como agente oxidante, por lo tanto, recibe electrones de un medio y se reduce[NELSON and BOLIN, 1995]. Durante el proceso de carga si se cuenta con una batería recargable (Fig. 1(b)) el proceso funciona de forma inversa, el agente oxidante se convierte en agente reductor y el reductor en agente oxidante, por lo que ahora el ánodo recibe los electrones y se carga la batería.

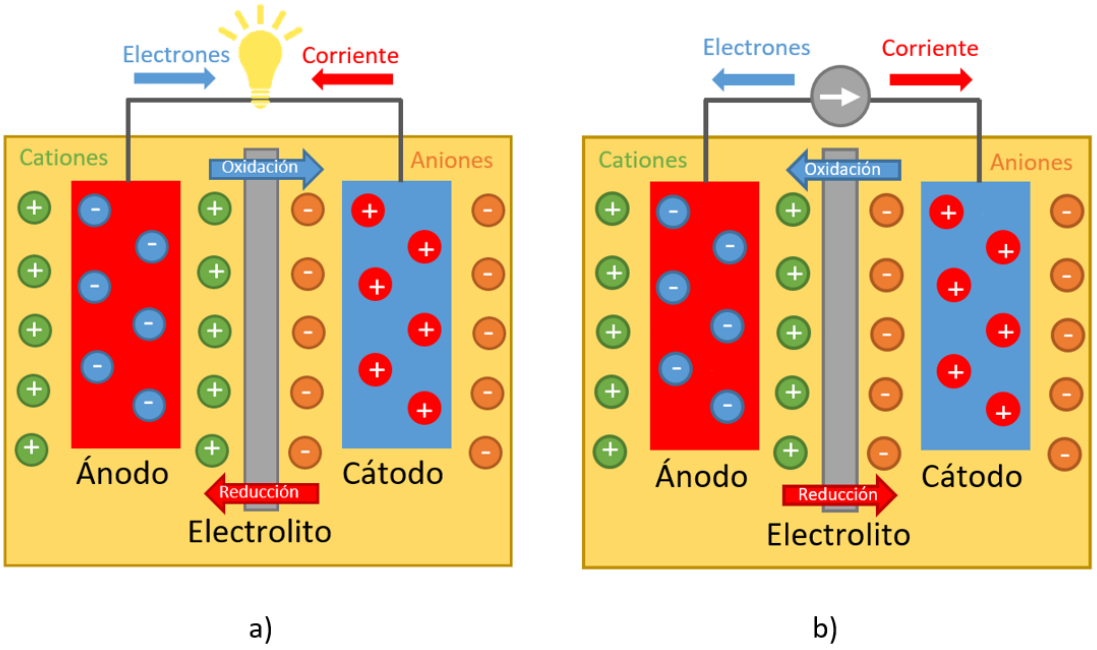

Figura 1: (a) Proceso redox durante descarga; (b) Proceso redox durante carga

5.1.3. Modelos de batería

Existen diversos modelos de baterías recargables que se utilizan en tópicos como sensores inalámbricos, automóviles eléctricos, teléfonos inteligentes, etc. Estos modelos tienen distinas clasificaciones, entre las más importantes relevantes para la tesis podemos encontrar las siguientes dos [Haverkort, 2009]:

- Modelos electromecánicos: Este modelo de baterías son los que se basan en procesos químicos para generar energía, algunos ejemplos son las baterías de iones de litio y polímero de litio (Fig. 2(a)).
- Modelos de circuitos eléctricos: Este modelo de baterías se compone de fuentes de voltaje y elementos pasivos como resistencias y capacitores (Fig. 2(b)). La composición del modelo es la siguiente:

  1. Un capacitor que representa la capacidad de la batería.
  2. Un circuito para descargar la batería.
  3. Una resistencia que representa la resistencia de la batería.

Los tipos de baterías más utilizados para dispositivos son basados en modelos electromecánicos, ya que tienen la ventaja de ser relativamente pequeñas y ligeras

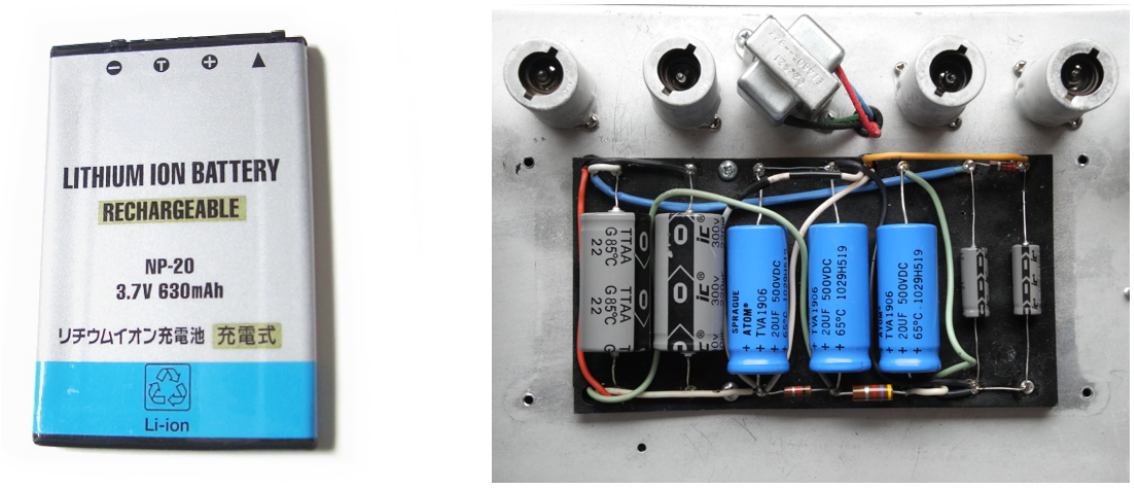

Figura 2: (a) Modelo electromecánico, batería de iones de litio; (b) Modelo de circuito eléctrico

en comparación con otros modelos, un ejemplo de como esta ventaja puede tener un gran impacto en el desempeño de un dispositivo es con los drones. Un dron es un dispositivo capaz de volar y maniobrar por el aire al que se le adapta una cámara para grabar durante el tiempo de vuelo, es necesario alimentar los motores de las hélices y la cámara con una batería que sea pequeña y liviana pero al mismo tiempo con mucha energía, si la batería es muy pesada el dron gastará más energía soportando ese peso provocando un desempeño ineficiente[Mallick et al., 2016]. Entre los tipos de baterías electromecánicas más comunes encontramos los siguientes:

5.1.3.1. Batería de ácido-plomo

Son el tipo de batería más utilizada en el mundo por los automóviles, su composición química es la siguiente:

- Ánodo: Plomo esponjoso.
- Cátodo: Dióxido de plomo.
- Electrolito: Ácido sulfúrico disuelto en agua.

La ventaja es que estas baterías tienen un tiempo de vida relativamente largo, sin embargo, esto depende completamente del uso. Entre las desventajas es que son muy

sensibles a las sobre descargas, es decir, que si se descarga completamente deja de funcionar debido a que sufre un proceso de sulfatación.

5.1.3.2. Batería de níquel-metal de hidruro (NiMH)

Este tipo de batería provee una solución a la necesidad de una tener una batería portable, que sea recargable y aparte más ligeras y pequeñas que sus predecesoras de níquel-cadmio(NiCd)[Gibbard, 1994]. Una de las principales desventajas de este tipo de baterías es que oponen mucha resistencia a la carga, a diferencia de otras baterías esta tiene que ser cargada con una corriente muy baja ya que de otra manera existe la posibilidad de que los elementos químicos reaccionen abruptamente produciendo una explosión.

5.1.3.3. Batería de iones de litio (Li-ion)

Las baterías de iones de litio (Li-ion)hoy en día son las más utilizadas en el mundo para generar una autonomía a los dispositivos y proveer energía, esto se debe a que tienen un desempeño muy bueno en relación con el tiempo de carga que proveen y la cantidad de ciclos de vida que resisten. Esta batería se puede conformar con diferentes tipos de electrodos (cobre, plata, etc.), sin embargo, lo importante es el electrolito que comparten los compuestos químicos como medio común que típicamente es de una sal de litio. Algunas de las características de batería son[Horiba, 2014]:

1. Alto voltaje por celda, aproximadamente de 3-4 V.
2. Alta eficiencia de energía.
3. El tiempo de vida de la batería es largo.

5.1.3.4. Batería de polímero de litio (LiPo)

Estas baterías están empezando a sustituir a las baterías Li-ion debido a que en vez de utilizar un electrolito liquido, es una combinación entre hexafluorofosfato de litio y disolventes orgánicos[Salameh and Kim, 2009]. Esta característica permite

que la batería pueda ser más flexible y ligera debido a que la consistencia es como un gel y no existe la posibilidad de que libere ácidos. Entre las principales ventajas encontramos una gran capacidad de energía, una larga vida o duración y muy ligeras[Ghossein et al., 2015], lo que lo hace ideales para drones, sensores y dispositivos de alta gama. También se cuenta con un circuito de protección que se encarga proteger la batería en caso de una sobrecarga, descargas y con un control de temperatura. Esta batería fue el que se selecciono para utilizarse debido a que a diferencia de otros tipos de batería, podemos cargar con más corriente y modificar libremente más parámetros sin poner en riesgo la vida de la batería.

5.1.4. Características de una batería LiPo

Cuando manejamos baterías de tipo LiPo tenemos que tener en cuenta que existen diversas características que nos dan información de como utilizarlas y cuales son las precauciones que debemos tener en cuenta para no provocar una explosión de la batería o en caso contrario, que deje de funcionar por una sobredescarga. Algunas de estas características podemos observarlas a simple vista en la batería, sin embargo hay unas que no, a continuación se muestran todas las características:

- Capacidad (Carga eléctrica almacenada): Desde un punto de vista especifico se puede definir como la cantidad de electrones que el ánodo de la batería puede contener, posteriormente, transformar estos electrones mediante un proceso químico en en energía eléctrica, (Fig 3). Si se desea explicar de una forma más general es la cantidad de energía eléctrica que se puede almacenar en la batería y suele medirse en Amperios hora (Ah), por ejemplo, si se tiene una batería de 5000 mA significa que puede ofrecer, teóricamente hasta 5000 mAh de forma constante por una hora.

- Tasa de descarga: Es la cantidad de energía a la que se puede descargar una batería de forma segura. También puede definirse como la cantidad de energía máxima que puede otorgar una batería por un periodo de tiempo determinado, suele indicarse con una Ç"que indica la capacidad de la batería y un numero, (Fig 3). La operación que se realiza para saber cual es el máximo de energía que esa batería puede proveer de forma segura es una simple operación aritmética

de multiplicación, por ejemplo, si la batería de 5000 mAh y la tasa de descarga indica 35C la operacion seria:

35C = 35 x (5000 mA) = 175,000 mA

- Tasa de carga: Aunque típicamente esta característica no se indica en la mayoría de las baterías, existe una limitante con respecto a la cantidad de energía a la que podemos cargar nuestra batería. Si la batería no indica esta característica debe cargarse a 1C, lo que equivale a cargarla máximo al tamaño de su capacidad, sin embargo, en algunos casos es posible cargar las baterías hasta 2C o 3C, algunos cargadores de carga rápida utilizan cargas de hasta 4C. Si contamos con una batería de 5000 mAh y queremos cargarla a 2C, podemos ajustar nuestro cargador a 10,000 mA valor que se obtiene de la siguiente operacion 2C = 2 x (5000 mA) = 10,000 Ah.

- Tensión o voltaje por celda: La tensión o voltaje por cerda tiende a ser estática y es de 3.7V, cuando esta completamente cargada puede subir hasta los 4.2 V y es importante no cargar más allá de ese valor ya que existe el riesgo de explosión o liberación de gases, por el caso contrario, no se debe descargar a menos de 3V ya que esto daña la batería permanentemente.

- Celdas de la batería: El numero de celdas se indica en la batería con un dígito que indica el numero de celdas seguido de la letra "S", por ejemplo, si la batería indica 3S se concluye que esa batería tiene 3 celdas conectadas en serie, (Fig 3). Los voltajes de estas celdas conectadas en serie se suman y por ende, el voltaje resultante aumenta:

  3S = (3.7 V) + (3.7 V) + (3.7 V) = 11.1 V

  El voltaje de esa batería sería de 11.1 V como se obtuvo en la operación anterior.

- Resistencia interna de la batería: La resistencia interna de una batería puede variar dependiendo del tiempo de uso y el desgaste de la batería, es importante tener en cuenta que esta resistencia aumenta conforme al uso y entre más grande sea, menos energía otorga.

- Vida de la batería: La vida de la batería tiende a medirse en ciclos de carga,

Figura 3: Ejemplo de una batería LiPo y sus características principales.

por lo regular la cantidad de ciclos que dura una batería de tipo LiPo oscila entre los 500 y los 1000 ciclos dependiendo completamente de los cuidados y uso. Se puede concluir que un uso eficiente de la carga y descarga de la batería se ve directamente reflejado en la vida de la batería.

5.1.5. Vida de una batería LiPo

La vida de una batería LiPo, como se explicó en la sección anterior, se mide en ciclos de carga. Estos ciclos de carga nos indican cuantas veces podemos descargar una batería y volverla a cargar sin que pierda sus propiedades electromotrices. La vida de la batería es un punto muy importante en el trabajo realizado ya que en ella se refleja la eficiencia del proceso de carga, en otras palabras, el método de carga tiene un impacto directo en el numero de ciclos de vida de la batería, por lo que entre más eficiente sea el método de carga más ciclos de vida otorgará la batería. La vida útil de la batería puede verse afectada por diversos factores como el tiempo de carga, un mayor tiempo de carga tiene un mayor impacto en la vida útil de la batería[Ayoub and Karami, 2015] ya que los químicos internos son sometidos durante un periodo más largo al flujo de corriente suministrados por el cargado,

también pueden afectar los parámetros de carga controlados por el método de carga y la temperatura de la batería.

5.2. Cargadores

Los cargadores juegan un rol importante en el proceso de carga de una batería ya que son los que controlan como se va a suministrar la energía. Todos los cargadores tienen una característica en común y es que, mediante un método de carga, son capaces de suministrar la energía de una forma en específico llamado método de carga, a su vez este método de carga ajusta los parámetros necesarios para que el proceso de carga sea más eficiente. Cada tipo de batería tiene sus limitantes en cuestión de parámetros de carga como la corriente máxima que se le puede suministrar, el tiempo de carga, el voltaje máximo que soporta la batería, etc. Estos parámetros se van a ver más a detalle la sección de métodos de carga. Existen dos clasificaciones de cargadores:

1. Cargadores convencionales: Este tipo de cargadores son los más comunes y utilizan un método de carga estandarizado.

2. Cargadores de carga rápida: Este tipo de cargadores implementan otros métodos de carga que si bien, no son un estándar entre los cargadores de carga rápida, permiten cargar más rápido una batería en comparación con el método de carga tradicional en los cargadores convencionales.

5.2.1. Definición

Un cargador es un dispositivo que se encarga de suministrar energía a una batería y utiliza un método de carga para controlar como es que se suministrara esa energía. El cargado debe tener una diferencia de potencial mayor al de la batería para que la corriente pueda fluir del cargador a la batería en sentido opuesto a la descarga. La configuración interna se compone de una bobina que reduce el voltaje de la conexión a la corriente eléctrica y unos rectificadores que la convierten de corriente alterna a corriente continua. Después de tener una corriente continua se procede modificarla de tal forma que se ajuste al método de carga que implementa el cargador.

5.2.2. Cargadores convencionales

Los cargadores convencionales implementan el método de carga de corriente-constante y voltaje-constante. Este método de carga divide el proceso de carga en tres etapas, el principal objetivo de este método de carga es proteger la batería de las sobrecargas y en caso de que la batería estuviera completamente descargada, comenzar a cargarla con una corriente muy pequeña para evitar el daño de la batería.

5.2.2.1. Método de carga de corriente-constante y voltaje-constante (CC-CV)

Este método de carga se divide en tres etapas y funciona para las baterías de Li-ion y LiPo[Lin et al., 2008]:

1. El Corriente por goteo (TC; del inglés Trickle Current): Si el voltaje de la batería está por debajo de un voltaje específico (típicamente 2.5V) el dispositivo empieza a operar en modo TC, en el cual la corriente suministrada por el dispositivo no es continua, si no por pulsos o por cortos periodos de tiempo. Esta etapa fue ideada para evitar el daño permanente de la batería, si una batería tiene poco voltaje y a continuación se le administra una gran cantidad de corriente, la celda se recupera de forma abrupta y va perdiendo su capacidad de retener electrones.

2. Corriente Continua (CC; del inglés Constant Current): Después de que la batería alcanza un voltaje mayor al voltaje específico para operar en este modo (típicamente 2.5V) el dispositivo comienza a suministrar corriente de manera continua, por lo que se almacena corriente rápidamente dentro de la batería.

3. Voltaje Constante (CV; del inglés Constant Voltage): Cuando el voltaje de la batería rebasa el voltaje específico para comenzar a operar en este modo (típicamente 4.1V) la corriente comienza a disminuir poco a poco hasta que el voltaje de la batería se establece en 4.2V. Una vez alcanzado este voltaje la batería está completamente cargada.

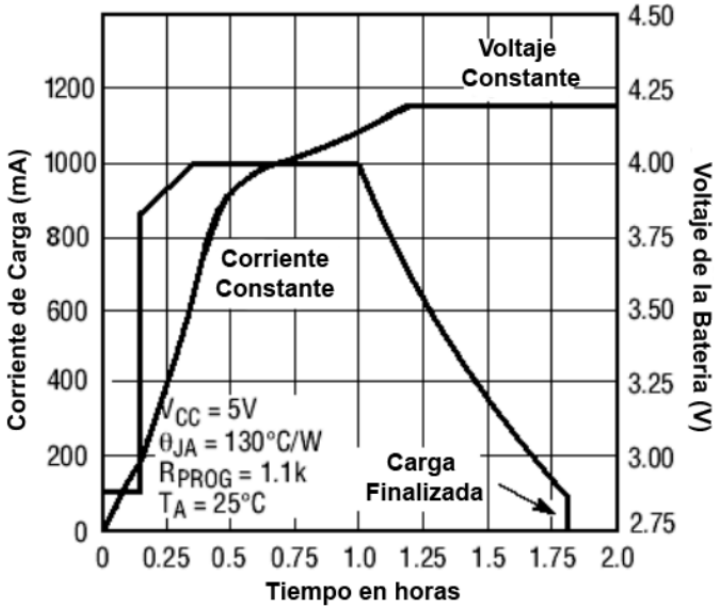

Figura 4: Ciclo de carga para una batería de 1000 mA con el método corriente-constante y voltaje-constante[Corp, 2015].

5.2.3. Cargadores de carga rápida

Los cargadores de carga rápida implementan métodos distintos al convencional, estos métodos pueden ser analíticos, basados en las propiedades de los componentes químicos de la batería o en algún tipo de técnica de optimización. Típicamente los cargadores convencionales utilizan una tensión de 5V y 1A o 2A de corriente, en el caso de los cargadores de carga rápida, el voltaje es dinámico, por lo que puede ir de los 5V a los 12 V y la corriente desde 1A a 3A. Los que se van a revisar en el documento son:

1. Método de carga multi-pasos a corriente constante.
2. Método de carga por pulsos.
3. Método BoostCharging.

5.2.3.1. Método de carga multi-pasos a corriente constante

Típicamente este método se aplica a las baterías de NiMH, sin embargo, es posible aplicar la metodología del método de carga a otro tipo de baterías como las de Li-ion

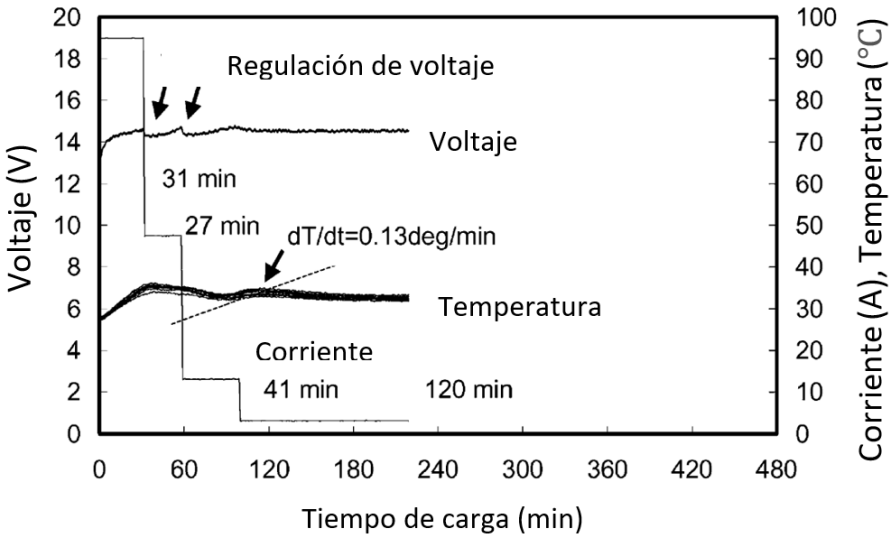

Figura 5: Ciclo de carga de una batería con el método de carga multi-pasos a corriente constante[Ikeya et al., 2002].

y LiPo. En el método se ajusta la corriente de forma automática dependiendo del voltaje de la batería, la idea principal es iniciar con una grande cantidad de corriente fluyendo hacia la batería, cuando la batería rebasa un cierto nivel de voltaje, esa corriente disminuye y así sucesivamente hasta alcanzar el nivel de voltaje máximo en la batería que indica que esta completamente cargada. Si se observa la corriente en una gráfica en relación con el tiempo de carga se puede observar un gráfico escalonado en donde dependiendo del nivel de voltaje, la corriente decae de forma abrupta.

5.2.3.2. Método de carga por pulsos

Este método es muy común en cargadores de carga rápida para baterías de Li-ion, sin embargo, es un poco caro replicar este método de carga en la experimentación debido a las bases de su funcionamiento. El principal objetivo que busca este método es distribuir de una manera uniforme los iones en el electrolito, reducir el tiempo de carga, disminuir la degradación de la batería y con esto, aumentar la vida útil de la batería. El método se basa en una frecuencia específica obtenida de la relación entre el tiempo de carga y la capacidad de la batería, este a sido el modelo adoptado por el método de carga ya que anteriormente esta frecuencia se obtenía a prueba y error

o de forma heurística. Esta frecuencia busca modificar el ciclo de trabajo "D", de tal forma que se consiga una distribución uniforme de los iones de litio dentro del electrolito y así reducir su degradación. Los pulsos se definen en base a la siguiente expresión[Ayoub and Karami, 2015]:

$$n = D * i$$

la densidad de corriente denotada con la variable i:

$$i = Fk(1-\theta)^{1-\alpha}\theta^{\alpha}c^{1-\alpha}$$

en donde:

1. k: Valor constante estándar para las reacciones heterogéneas.
2. F: Faraday (96487C).
3. θ: Fracción molar.
4. α: Coeficiente de transferencia.
5. c: Concentración en la superficie del electrodo

5.2.3.3. Método de carga BoostCharging

El método de carga BoostCharging esta basado en la observación, se define como un algoritmo de carga súper rápida para baterías de Li-ion y como principal característica de este método de carga es que las baterías completamente descargadas pueden ser cargadas a altas corrientes sin tener daños considerables a la vida de la batería. En la búsqueda de acelerar el proceso de carga, este método se puede implementar en la carga de baterías Li-ion para pasar a CV (Voltaje constante) en un lapso de tiempo muy corto, entonces, la modificación en el proceso de carga de este método es en la etapa de CC (Corriente constante). Durante la etapa de CC (Corriente constante) en donde la corriente que otorga el cargador es bastante considerable y por ende, el proceso de carga pasa casi inmediatamente a la siguiente etapa, CV (Voltaje constante). La corriente puede aumentar hasta 4.5C durante la etapa de CC (Corriente constante), por ejemplo, si se quiere cargar una batería de

Tabla 1: Comparación entre la capacidad de la batería resultante durante los ciclos de carga aplicando el método de carga BoostCharging y CC-CV[Notten et al., 2005].

Ciclos de vida	Capacidad de la batería		
	Método Booscharging: $I^{max} = 4.5C$ $V^{max} = 4.3V$	Método Booscharging: $I^{max} = 4.5C$ V^{max}=4.2V	Método CC-CV
0	100 %	100 %	100 %
100	78 %	92 %	97 %
200	45 %	82 %	90 %
300	35 %	65 %	81 %
400	20 %	55 %	65 %

1,000 mAh, la corriente del cargador puede aumentar hasta los 4,500 mA; durante la etapa de voltaje constante, el voltaje se mantiene en los 4.2V como lo especifíca el método de carga CC-CV. Existe cierto impacto en la vida de la batería al utilizar este método (Tabla 1), esto se debe a que aunque no sea perjudicial para la batería utilizar grandes parámetros de carga en comparación con los que se usan en un método de carga tradicional, el forzar el proceso químico redox afecta la capacidad de retención de electrones por parte del electrodo negativo en la batería.

5.3. Técnicas metaheurísticas

La metaheurística se define como procedimientos heurísticos de alto nivel que buscan resolver problemás de optimización improvisando técnicas y algoritmos en su mayoría inspirados en la naturaleza. Las técnicas metaheurísticas nacen como una alternativa a las técnicas convencionales buscando reducir el tiempo de computo de los problemás de optimización, algunas veces estas técnicas no llegan al resultado óptimo, sin embargo, tienen una gran aproximación y se utilizan cuando existe cierto grado de incertidumbre en el fenómeno[Bianchi et al., 2009]. Técnicas metaheurísticas como optimización basada en colonia de hormigas(ACO; del inglés Ant Colony Optimization), cómputo evolutivo(EC; del inglés Evolutionary Computing) y Optimización de enjambre de partículas(PSO; del ingles Particle Swarm Optimization) se revisan a continuación.

5.3.1. Ant Colony Optimization (ACO)

Esta técnica metaheurística para resolver problemás de optimización basada en la naturaleza es de las más eficientes. Esta técnica se desarrolló en base a la observación y se inspira en el comportamiento de las hormigas y en su biología; cuando las hormigas buscan caminos desde su nido hasta una fuente de alimento, no lo hacen de forma aleatoria, parecen buscar el camino más corto y en el que tarden menos tiempo en llegar, entonces este comportamiento se puede utilizar para resolver problemás de optimización. Cuando las hormigas caminan van dejando un compuesto químico llamado feromona, esta se concentra y se hace más fuerte cuando las hormigas detectan un camino a seguir que los guía a una fuente de comida. Si se encuentran con dos caminos, uno largo y uno corto las hormigas escogen de forma aleatoria cual seguir, en caso de ser el largo cuando estas regresan al nido, algunas se irán por el camino corto y comenzarán a soltar la feromona que se irá concentrando y haciendo cada vez más fuerte, lo que obliga a las demás a seguir ese camino. Existen tres principales procedimientos que caracterizan a esta técnica[Bianchi et al., 2009]:

1. ConstructAntSolutions: Es el procedimiento en el que hormigas artificiales crean caminos estocásticamente. Para una determinada hormiga existe un índice de probabilidad de llegar de un nodo hasta un nodo satisfactorio, esta probabilidad se ajusta con una función incremental que recibe parámetros como donde se encuentra feromona en el suelo y un valor heurístico.

2. EvaporatePheromone: Es el proceso en donde la feromona esparcida por las hormigas disminuye del camino. Es importante disminuir la concentración de feromonas del camino para evitar que el algoritmo converja rápidamente en un mínimo local, a este proceso se le llama actualización local.

3. DeamonActions: Este proceso evalúa y compara mediante una función objetivo las soluciones aportadas por las hormigas aplicando heurísticas de búsqueda local y cuando determina la mejor solución incrementa la cantidad de feromona, este incremento de feromona se le conoce como actualización global.

5.3.2. Evolutionary Computing (EC)

La computación evolutiva es una técnica metaheurística que se basa en la naturaleza, específicamente en el proceso de la evolución. El paradigma neo darwinista es el más aceptado y contempla procesos biológicos como la selección, competencia, reproducción y mutación. A cada posible solución que se evalúa conforme a una función de aptitud se le conoce como individuo y a un conjunto de individuos como una población; durante el proceso de selección se identifica y escoge a los mejores individuos que después compiten para después reproducirse[Fogel, 2004]. Cada individuo tiene una estructura genética que lo hace diferente del resto y al momento de reproducirse pasa a su descendencia sus mejores características creando una nueva población de individuos, cada vez que surge una nueva población a partir de otra se le conoce como una nueva generación. El proceso de mutación se utiliza para generar pequeños cambios aleatorios a los individuos que eviten que el algoritmo converja en un mínimo local.

5.3.3. Particle Swarm Optimization (PSO)

Esta técnica metaheurística también se basa en la naturaleza y busca simular el vuelo de una parvada de aves. En un PSO un conjunto de partículas se crean en el campo de búsqueda de un problema de optimización y cada partícula ejecuta una función de objetivo de su posición actual. Después determina el próximo movimiento que hará en base a un historial de sus últimos movimientos y las mejores posiciones en el espacio de búsqueda en el que a estado con una o más partículas del enjambre.Una vez que todas las partículas hayan hecho sus movimientos, se procede a una nueva iteración.

5.4. Algoritmos Genéticos

En la era digital el uso de algoritmos genéticos no es un tema nuevo, en 1970 J. H. Holland hizo una propuesta significativa para la ciencia y la ingeniería. Esta contribución propone resolver problemás de optimización mediante el uso de algoritmos heurísticos basados en la búsqueda de posibles soluciones mediante la adaptación de teorías biológicas[Man et al., 1996]. Un algoritmo genético ofrece, en la mayoría de los casos, un resultado muy aproximado a la solución de un problema, no se requiere un conocimiento amplio de un tema para su implementación a diferencia de otros algoritmos de optimización basados en grandiente que utilizan una ecuación derivable con las propiedades del fenómeno, en conclusión, la función objetivo de un algoritmo genético no necesita ser tan compleja en comparación con estos algoritmos basados en gradiente.

5.4.1. Introducción a los algoritmos genéticos

El computo evolutivo es una rama de la inteligencia artificial que se basa en los procesos de la evolución natural. Una de las vertientes del cómputo evolutivo son los algoritmos genéticos, los cuales mediante la implementación de algoritmos heurísticos que se basan en la teoría de la evolución neo darwinista; reproducción, mutación, competencia y selección, buscan resolver problemás de optimización[Negnevitsky, 2005] [Konak et al., 2006]. Los algoritmos genéticos se inspiran en un proceso biológico en donde los individuos más fuertes son los ganadores en una constante competencia dentro de un contexto definido. Cada individuo representa una posible solución al problema dentro de un espacio de solución conocido como fenotipo. Un fenotipo en términos biológicos se define como un espacio de solución de un fenómeno observable conformado por atributos relevantes del fenómeno, para resolver el problema mediante algoritmos genéticos es necesario transformar el espacio del fenotipo a un espacio llamado genotipo, en el cual se realizan las operaciones involucradas en un algoritmo genético (Fig. 6), para que estas se lleven a cabo se requiere una representación computable, donde típicamente se utiliza una representación binaria.

Cada una de las posibles combinaciones de valores dentro del espacio del geno-

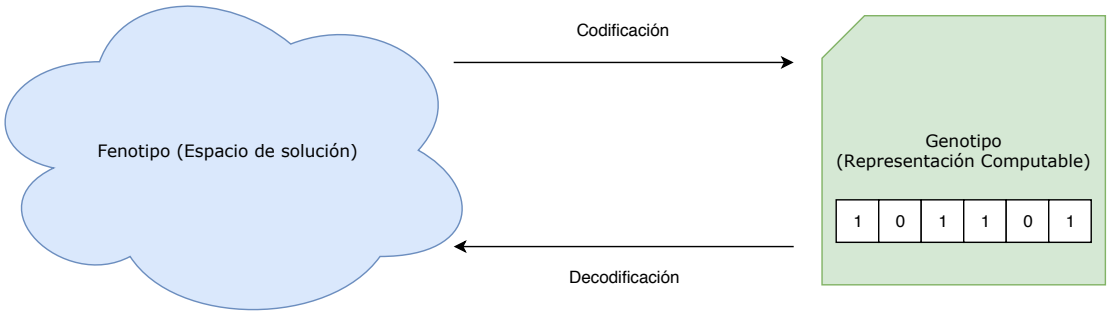

Figura 6: Representación de los espacios fenotipo y genotipo.

tipo representa un individuo (también llamado cromosoma), la expresión de este individuo es una cadena de genes que se representa en bits (1,0), en donde cada bit o conjunto de bits son asociados a una característica de la solución esperada (Fig. 7).

Se le conoce como población a un conjunto de individuos y representa un espacio de búsqueda en donde cada individuo es candidato a representar una posible solución al problema[Goldberg, 1989]. Es posible calificar la eficiencia de la población en base al promedio obtenido de una función de aptitud que evalúa cada individuo (Fig.8), puntuando que tan cerca se encuentra del objetivo o solución. De esta forma podemos expresar de una forma cuantificable que tan eficiente es cada individuo o bien, la población entera. Después de evaluar a los individuos existen diversas formás para seleccionar los que se van a reproducir y así, generar otros nuevos que sustituirán a la población, esto es importante ya que durante el proceso de reproducción los individuos seleccionados pasan, mediante operaciones de cruce de genes (Fig.8), parte de su estructura genética a la nueva generación, esperando que la estructura

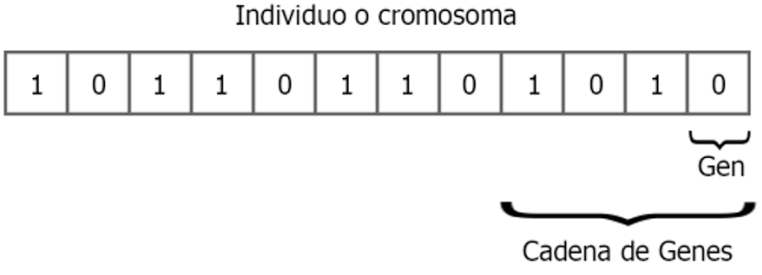

Figura 7: Estructura de un individuo.

genética del nuevo individuo sea mejor que la de sus predecesores. Para que la población se diversifique y el algoritmo no converge en un mínimo local es necesario agregar un operador de mutación a cada individuo (Fig.8), esto permite buscar de forma aleatoria en otros lugares dentro del espacio de la solución[Lin, 2009], el porcentaje de mutación tiende a ser muy pequeño puesto que un cambio significativo al individuo convertiría la búsqueda heurística en una búsqueda aleatoria. Una vez que la descendencia ya mutó, es posible reemplazar la anterior generación con la nueva población de individuos y mediante un proceso iterativo estas operaciones siguen efectuándose hasta finalizar el algoritmo genético. Existen dos formás de finalizar un algoritmo genético:

1. Cuando alguno de los individuos alcanzo los requerimientos establecidos por la función de aptitud que se llego a un resultado aceptable.

2. Cuando el algoritmo genético ya alcanzo un numero de generaciones determinadas por el usuario que puede ser en base a la experiencia o análisis previos.

La estructura de los pasos de un algoritmo genético (Fig.8) se describen en las secciones posteriores.

5.4.2. Población inicial

Existen dos formás de crear una población inicial, en el primero de los casos podemos crearla de forma a priori, esto significa que los individuos se generan a partir de un conocimiento previo con características (genes) establecidas por el usuario; la segunda forma de crear una población inicial es generándola aleatoriamente, este es de los casos más comunes en algoritmos genéticos debido a que no se tiene un conocimiento del comportamiento de la población y por ende, es muy probable que converja en un resultado satisfactorio al tener un mayor campo de búsqueda en el espacio de solución.

5.4.3. Función de aptitud

La función de aptitud es la función que el algoritmo trata de optimizar [Mitchell, 2013]. La palabra aptitud es un termino que adopta algoritmos genéticos

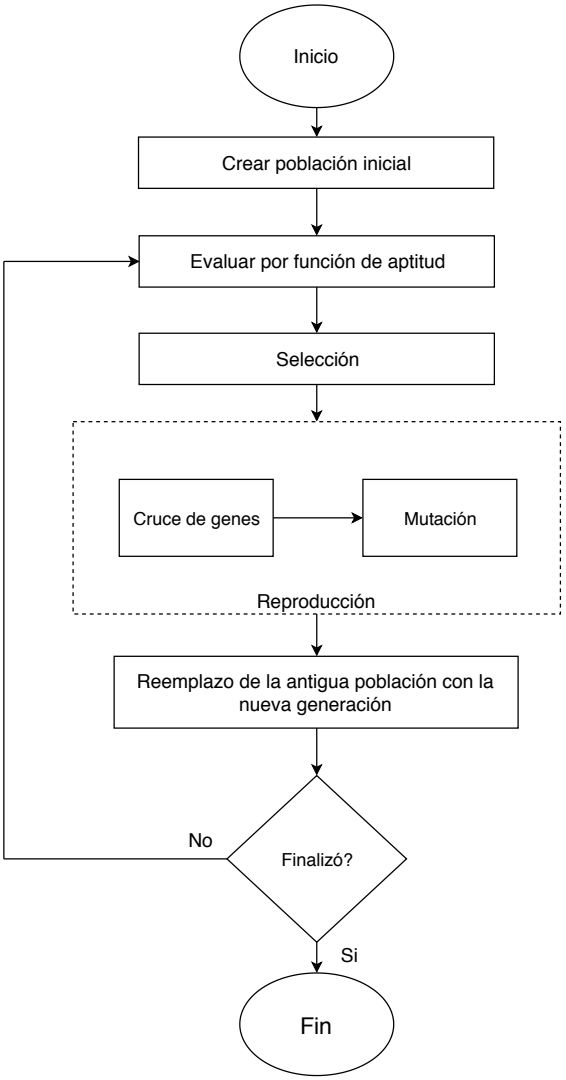

Figura 8: Diagrama de procesos de un algoritmo genético.

de la teoría de la evolución y significa que tan adaptado se encuentra un individuo en relación con la solución o el resultado que se espera. Se podría considerar que es la parte más importante de un algoritmo genético, ya que es la única etapa del algoritmo que determina que cambió en los individuos a través de las generaciones, la importancia recae en elaborar una función de aptitud acorde al fenómeno a optimizar, la función de aptitud puede determinar si el algoritmo tendrá resultados positivos o no tener ninguno[Heiss-Czedik, 2009]. Existen dos formás de interpretar los resultados de una función de aptitud:

1. Maximización: En base al nivel de adaptación de un individuo después de ser evaluado por una función de aptitud, se determina que el individuo es más

apto al obtener una mayor resultado.

2. Minimización: Se considera que un individuo es más apto al obtener un menor resultado después de ser evaluado por la función de aptitud, en la mayoría de los casos el cero funge como el mayor nivel de adaptación por lo que entre más se acerque el resultado de la función de aptitud al cero se considera más apto.

5.4.4. Selección

El proceso de selección, en la mayoría de los casos, identifica los individuos mejor adaptados en base a la puntuación resultante de la función de aptitud, una vez identificados se seleccionan mediante alguna técnica de selección y pasan a la etapa de reproducción. Existen diversos métodos de selección entre los que encontramos selección por torneo, selección proporcional de la rueda de la ruleta (PRWS; del inglés Proportionate Rou-lette Wheel Selection) y muestreo estocástico universal (SUS; del inglés Stochastic Universal Sampling).

5.4.4.1. Selección por torneo

Este método de selección es el más común en algoritmos genéticos debido a la facilidad de implementación y su eficiencia. De esta técnica se seleccionan de entre toda población n individuos de forma aleatoria, una vez seleccionados compiten entre ellos mismos y gana el que tiene un mayor valor de aptitud[Goldberg and Deb, 1991]. Típicamente el tamaño de individuos seleccionados de forma aleatoria para el torneo es de 2, sin embargo existen otras variantes de esta técnica en donde ese tamaño puede ser más grande (Fig. 9).

5.4.4.2. Selección proporcional de la rueda de la ruleta (PRWS)

Posteriormente a este método de selección cada individuo debe tener un valor de aptitud. La probabilidad de selección de los individuos es proporcional al valor de aptitud de cada uno, en otras palabras, entre un mayor valor de aptitud existe una mayor probabilidad de que el individuo sea seleccionado. El promedio de la

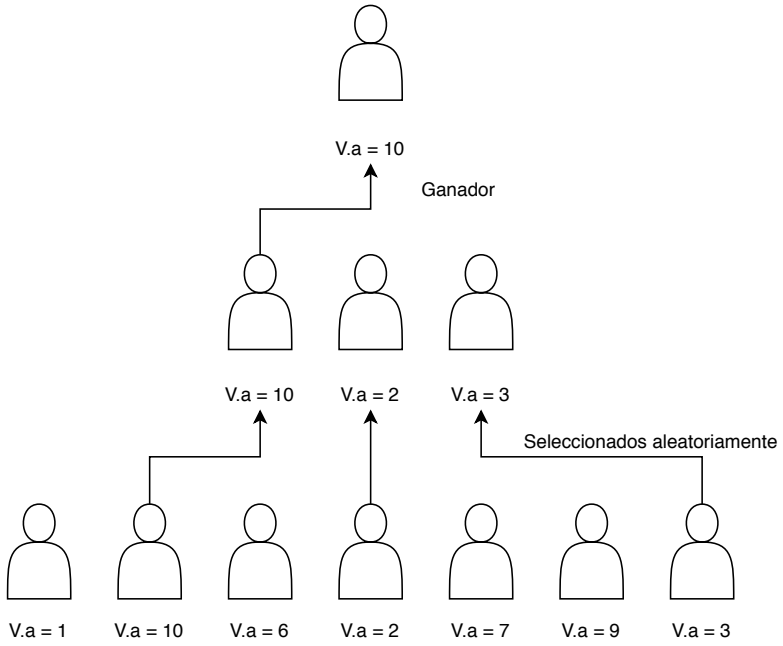

Figura 9: Representación gráfica del método de selección por torneo por valor de aptitud (V.a.).

sumatoria de los valores de aptitud de la población indica la magnitud total de selección y se representa con una ruleta conformada por secciones las cuales a su vez representan los valores de aptitud de cada individuo (Fig 10).

Si se cuenta con una población de n (n es el numero de individuos en la población), $P = \{x_1, x_2, x_3, ..., x_n\}$ en donde x_i es el valor de adaptación de $f(x_i)$, entonces la probabilidad de selección $ps(x_i)$ se da por[Jinghui Zhong et al., 2006]:

$$ps(x_i) = \frac{f(x_i)}{\sum_{i=0}^{n} f(x_j)}, j = 1, 2, 3, ..., n$$

5.4.4.3. Selección por muestreo estocástico universal (SUS)

Este método de selección fue propuesto por Baker J. y se considera una variante del método de selección proporcional de la rueda de la ruleta (PRWS), la principal diferencia de este método es que en vez de utilizar un solo punto fijo se utilizan N puntos fijos, representando el numero de individuos que se van a seleccionar en un solo giro de la rueda de la ruleta. La distancia entre cada punto fijo se determina con $\frac{1}{N}$[Pencheva et al., 2009] y la posición del primer punto fijo se determina de forma aleatoria, a partir de ahi la distancia entre cada punto fijo es de. Si queremos determinar 2 puntos fijos, determinamos su separación $\frac{1}{2} = 0.50$ para una población

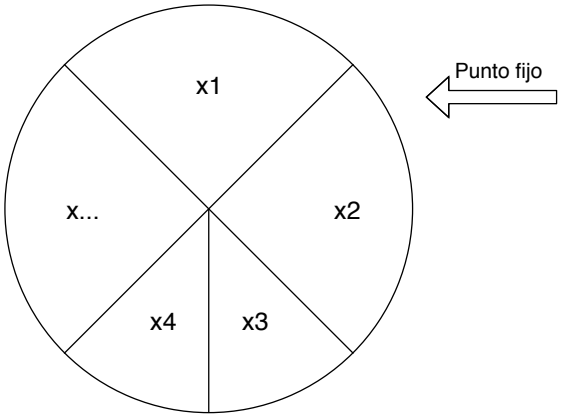

Figura 10: Representación gráfica del método de selección PRWS.

$P = \{x_1, x_2, x_3, ..., x_n\}$ hasta $n = 5$(Fig. 11).

5.4.5. Cruce de genes

Las operaciones de cruce en algoritmos genéticos tiene un gran peso en el valor de aptitud de las próximás generaciones, ya que durante este proceso los individuos seleccionados para la reproducción generan descendencia a partir de la combinación de las características (genes o cadena de genes) de ellos mismos, en donde la reproducción de dos individuos generan 2 descendientes[Guzhan et al., 1998]. Entre las técnicas más comunes encontramos el cruce en un punto, cruce en dos puntos y cruce en multi-puntos.

5.4.5.1. Técnica de cruce en un punto

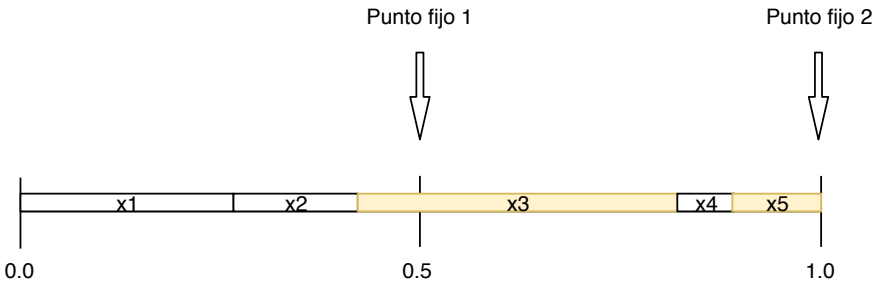

Figura 11: Representación gráfica del método de selección SUS.

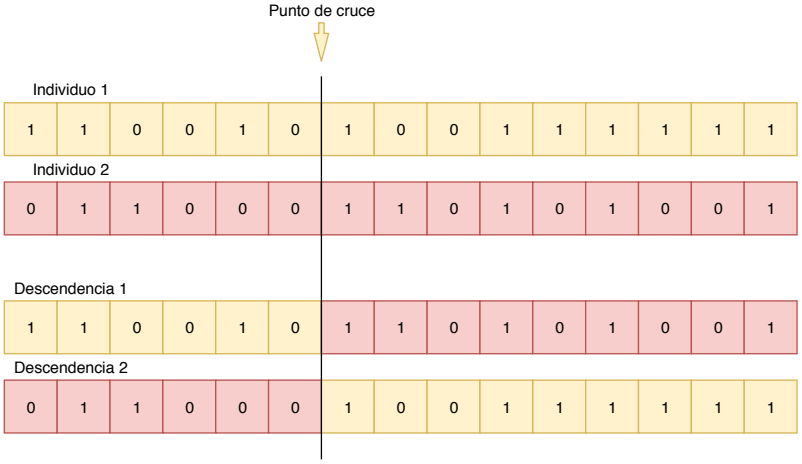

Figura 12: Técnica de cruce en un punto aleatorio.

Esta técnica es la más sencilla de implementar en cuanto a técnicas de cruce se refiere, el individuo tiene una estructura genética cuando hablamos del fenotipo, para su cómputo es necesario que esa estructura genética sea una cadena de bits (o genes), en donde uno o varios bits representan una característica especifica. La técnica selecciona un punto aleatorio en la estructura genética de los dos individuos y a partir de ese corte se intercambian las cadenas de bits (Fig. 12).

5.4.5.2. Técnica de cruce en un dos puntos

La técnica de cruce de dos puntos funciona de la misma manera que la técnica de cruce de un solo punto, la principal diferencia es que en vez de seleccionar un punto aleatorio entre los individuos, se seleccionan 2 puntos y después se hace el intercambio de las cadenas de bits (Fig. 13). En esta técnica existe una mayor posibilidad de intercambiar las mejores características de cada individuo al tener más puntos intercambio de bits, en un estudio se determino en base a la comparación entre esta y otras técnicas ser la más eficiente [Guzhan et al., 1998].

5.4.5.3. Técnica de cruce en multi-puntos

Esta técnica no es muy diferente a la de un punto y la de dos puntos de cruce, la diferencia es que $n_c \geq 3$ en donde n_c es el número de cortes. El cruce entre las cadenas de bits sera de $n_c + 1$ y el intercambio de los bits se hace de forma secuencial

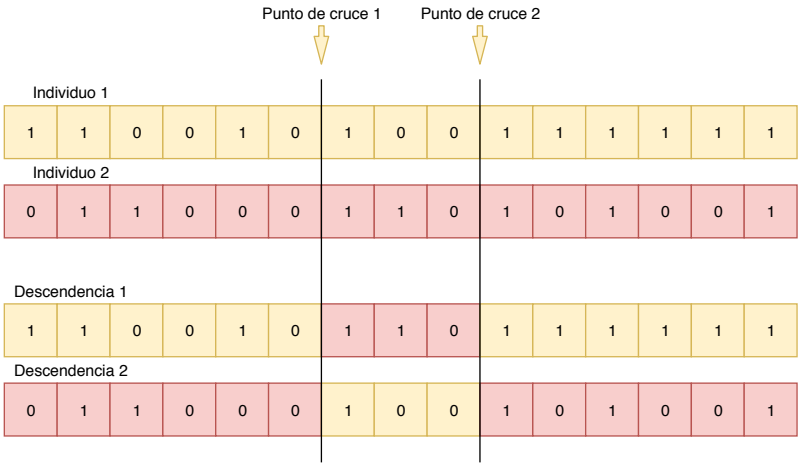

Figura 13: Técnica de cruce en dos puntos aleatorios.

(Fig. 14). Esta técnica tiende a ser muy útil cuando el tamaño del individuo en su estructura genética es muy grande, de caso contrario el resultado de la búsqueda tiende a ser muy aleatorio debido a la gran variación de los genes.

5.4.6. Mutación

El proceso de mutación es muy importante para expandir la búsqueda en el campo de solución al proveer datos aleatorios dentro de la estructura genética de los individuos. Al expandir el campo de búsqueda se reduce el riesgo de que el algoritmo converja y se estanque en un mínimo local. El operador selecciona de manera aleatoria uno o varios bits de la estructura genética de un individuo y lo

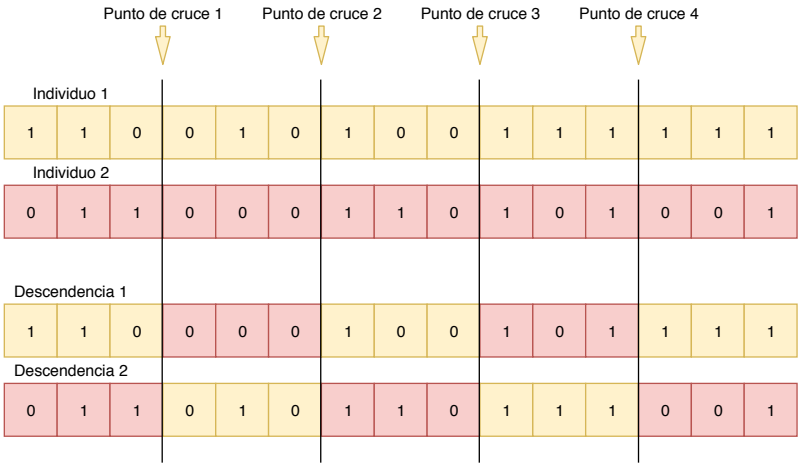

Figura 14: Técnica de cruce en multi-puntos aleatorios.

cambia (Fig. 15), la probabilidad de mutación es determinada por el programador, sin embargo, típicamente se utilizan probabilidades de mutación muy bajos para evitar que el algoritmo tienda a ser muy aleatorio.

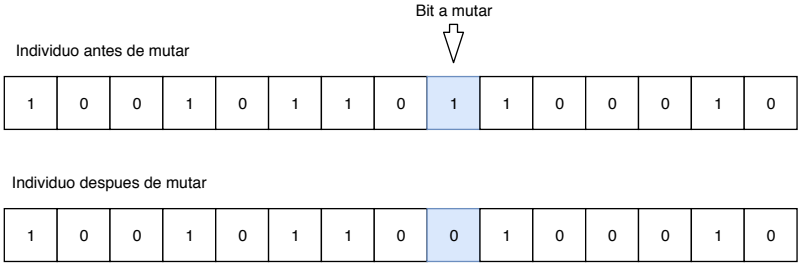

Figura 15: Ejemplo de mutación.

6. Desarrollo

Con el fin de seleccionar el tipo batería y el método de carga convenientes para resolver la hipótesis planteada, se desarrollo un prototipo previo a la solución final. Este prototipo constaba de utilizar un dispositivo integrado que es el que mayormente se utiliza para cargar baterías de tipo LiPo y modificar su proceso de carga para adaptar un método de carga diferente.

6.1. Prototipo inicial

El objetivo de este prototipo es establecer qué método de carga utilizar para después implementar una técnica de optimización, así determinar los parámetros de carga óptimos que mejoren el proceso de carga, los métodos analizados fueron el método de carga por pulsos y el método de carga multi-pasos, permitiendo la propuesta e implementación de un nuevo método basado en los anteriores mencionados. Los parámetros de carga se establecieron de forma empírica y se dividió el prototipo en 3 módulos:

1. Módulo de control.
2. Módulo de carga.
3. Módulo de baterías.

El prototipo es capaz de cargar 4 baterías de tipo LiPo simultáneamente y ejecuta el método de carga con parámetros distintos en cada una de ellas, sin embargo, es necesario descargar las 4 baterías manualmente por cada ejecución del experimento, esto se debe a que es necesario que las baterías estén completamente descargadas al momento de iniciar con la ejecución del experimento.

6.1.1. Módulo de control

Se encarga de generar las señales de control para los diferentes dispositivos, así como recolectar información de las baterías como su nivel de voltaje. Se conforma por un Arduino Mega 2560 y un conjunto de relevadores. El arduino se conecta a los

relevadores, potenciómetros digitales y a las baterías, en el primero de los casos es para controlar hacia donde se conecta la batería. El relevador es capaz de conectar la batería al circuito de carga y al ADC(Convertidor Análogo-Digital; del ingles Analog Digital Converter) del arduino. Las acciones de este módulo son la siguientes:

1. Ejecutar el método de carga propuesto en cada una de las baterías, modificando el valor de los los 4 potenciómetros digitales conectados los dispositivos TP4056 [Corp, 2015] en el pin PROG bajo el protocolo de comunicación SPI(Bus de interfaz de periféricos serie; del inglés Serial Peripheral Interface) para los parámetros de corriente.

2. Llevar un control de tiempo muy especifico para ejecutar correctamente el método de carga en cada uno de los cargadores TP4056 para los parámetros de tiempo.

3. Leer el voltaje de las baterías permitiendo la conexión de la batería al ADC del arduino mediante el control de los relevadores.

4. Controlar la conexión de la batería cambiando el estado del relevador, esta conexión puede ser batería a circuito de carga y batería a ADC.

Estas acciones están coordinadas de tal forma que el módulo ajusta el método de carga dependiendo de sus parámetros de corriente y tiempo mientras cada 5 segundos lee el voltaje de la batería, esta ultima acción requiere el control de los relevadores.

6.1.2. Módulo de carga

En este módulo se encuentran dos dispositivos funcionando de forma paralela por cada batería conectada, el primero es el dispositivo de carga TP4056 y el segundo es un potenciómetro digital. El objetivo de este módulo es suministrar la energía para cargar las baterías y se compone de 4 dispositivos TP4056 y 4 potenciómetros digitales MCP4131 de 10K ohms.
El dispositivo TP4056A utiliza el método de carga convencional CC-CV y es para baterías de tipo Li-ion y LiPo de una celda, el microcontrolador cuenta con un pin

Tabla 2: Valores de configuración para RPROG.

RPROG (Kohm)	I^{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

PROG [Corp, 2015] el cual modula la corriente de salida del dispositivo hacia la batería gracias a una resistencia de referencia RPROG (Tabla 2).

Como se observa en la tabla 2 es posible cambiar la corriente que proporciona el dispositivo al modificar la resistencia conectada a el, en este caso el potenciómetro digital. El potenciómetro digital cambia el valor de su resistencia modificando sus registros internos y estos se modifican mediante una conexión bajo el protocolo de comunicación SPI. El protocolo de comunicación SPI es síncrono y se realiza mediante 4 conexiones[Leens, 2009]:

- SCLK(Reloj): Manda los pulsos de reloj desde el máster bus todos los slave bus.
- MOSI: Línea de datos.
- MISO: Línea de datos.
- SS/SELECT: Linea para seleccionar que slave bus va a tener comunicación con el máster bus.

En este caso el arduino funciona como SPI máster y los potenciómetros digitales como los SPI slaves (Fig. 16).

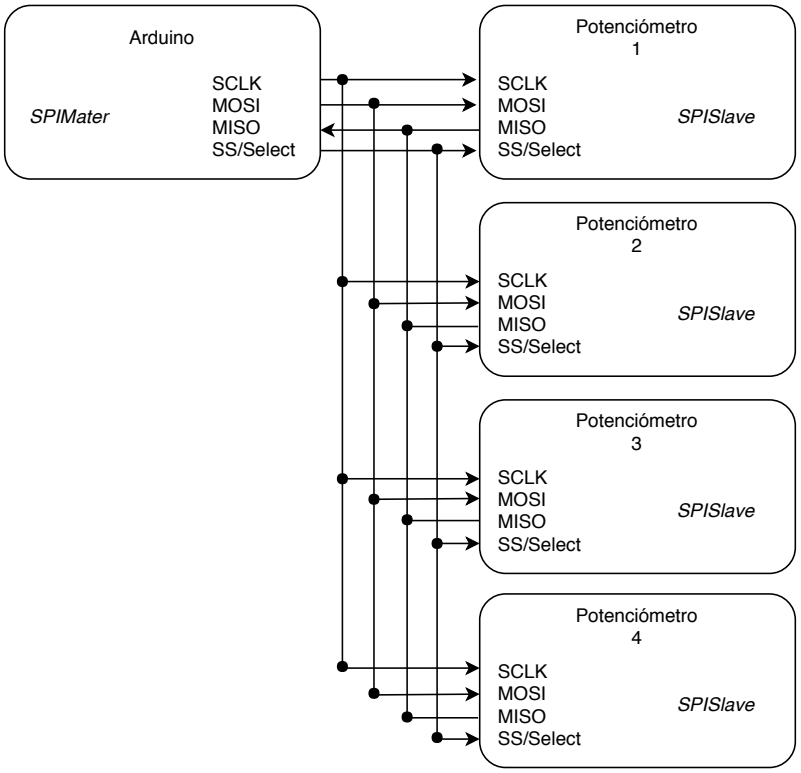

Figura 16: Conexión del arduino a los potenciómetros digitales MCP4131 [MicroChip, 2007].

6.1.3. Módulo de baterías

En este módulo se encuentran las baterías, las cuales proporcionan los datos necesarios para analizar la eficiencia de los parámetros del método de carga. Este módulo se compone de 4 baterías de tipo LiPo de 1000 mA. La principal razón por la que se tomo la decisión de utilizar baterías de tipo LiPo después de investigar y analizar las ya existentes es debido a su seguridad y adaptabilidad, a diferencia de su predecesor, la batería de Li-ion, un daño o avería en en esta batería podría ocasionar la liberación de ácidos que a diferencia de una batería LiPo, esta no libera ácidos al sobrecargares por su composición química, además de que las baterías de tipo LiPo cuentan con un circuito de protección que evitan precisamente un accidente de este tipo al desconectar la batería del proceso de carga si ya se encuentra a un limite de carga peligroso. También es posible cargar estas baterías a grandes corrientes superando hasta 4 veces su capacidad de carga, en comparación con las baterías NiMH las cuales ofrecen bastante resistencia a la corriente eléctrica.

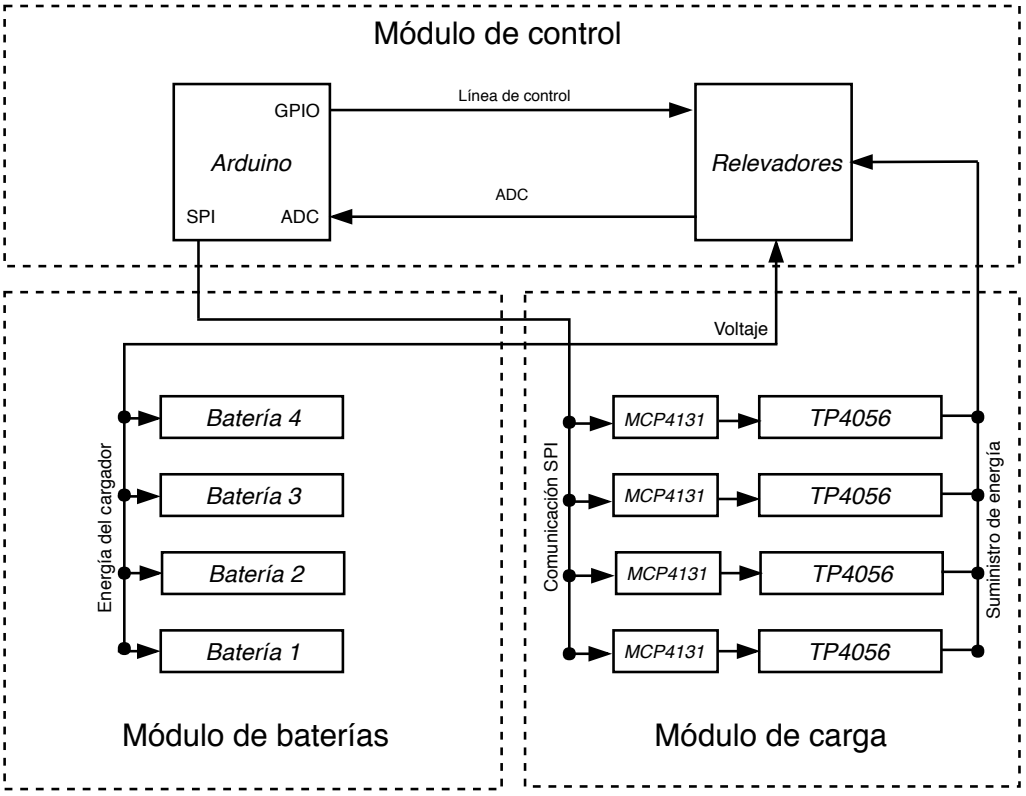

Figura 17: Diagrama del funcionamiento general del prototipo inicial.

6.1.4. Arquitectura y diseño

Cada módulo se desarrolló para funcionar de manera sincronizada, mientras el módulo de control se encarga de generar las señales necesarias para el funcionamiento del sistema, el módulo de baterías provee la información necesaria para determinar la eficiencia de los parámetros de carga implementados por el módulo de carga mediante la lectura del voltaje de la batería, el cual es analizado y registrado por el sistema de control (Fig. 17).

La lectura del voltaje es la parte más importante del experimento, esta acción se realiza cada 5 segundos y se ejecuta con el apoyo de los relevadores. Los relevadores desconectan la batería del módulo de carga, específicamente del dispositivo TP4056, esto es necesario para leer correctamente el voltaje de la batería ya que si se hace una medición del voltaje con la batería conectada al dispositivo de carga, el voltaje capturado seria el voltaje que proporciona el cargador y no el de la batería, esto se debe a que el cargador debe tener un mayor voltaje que el de la batería para que la energía fluya en el sentido correcto. Como se menciono anteriormente, los parámetros

de carga para el método de carga propuesto se generaron de forma empírica, esto se debe a que era importante probar que el método de carga prepuesto era más eficiente con respecto a un menor tiempo de carga. El método de carga propuesto se explica en una sección posterior.

6.2. Método de carga

El método de carga propuesto es una combinación entre el método de carga por pulsos y el método de carga multi-pasos, la principal característica de este método de carga es metodología en que se implementan los parámetros de carga, estos son los parámetros de corriente y los parámetro de tiempo. Este método de carga ajusta la corriente de salida del cargador a un valor dado por el parámetro de corriente y este se prolonga el tiempo establecido por el parámetro de tiempo, por ejemplo, se cuenta con 2 parámetros de corriente y 2 parámetros de tiempo en donde:

- Parámetros de corriente: 1000 mA y 500 mA.
- Parámetros de tiempo: 1 seg y 3.4 seg.

Con la configuración anterior el método de carga ajusta la salida de corriente del cargador a 1000 mA (primer parámetro de corriente) durante un lapso de tiempo de 1 seg (primer parámetro de tiempo), posteriormente, ajusta la salida de corriente del cargador a 500 mA (segundo parámetro de corriente) durante un lapso de tiempo de 3.4 seg (segundo parámetro de tiempo). Esto sucede de forma iterativa hasta que la batería alcanza un voltaje de 4.2 V. Las características que se combinaron del método de carga por pulsos y el método de carga multi-pasos en el primero de los casos, es el proceso de oscilar la corriente de un parámetro a otro, como por ejemplo en el caso anterior de 1000 mA a 500 mA, estos parámetros se prolongan por tiempos definidos al igual que el método de carga multi-pasos. Este método de carga tiene dos esquemas de carga, estos son, corriente constante a 2 pasos (CC 2 steps) y corriente constante a 3 pasos (CC 3 steps):

1. CC 2 Steps: Este esquema de carga percibe 2 parámetros de corriente y 2 parámetros de tiempo (Fig. 18 (1)).

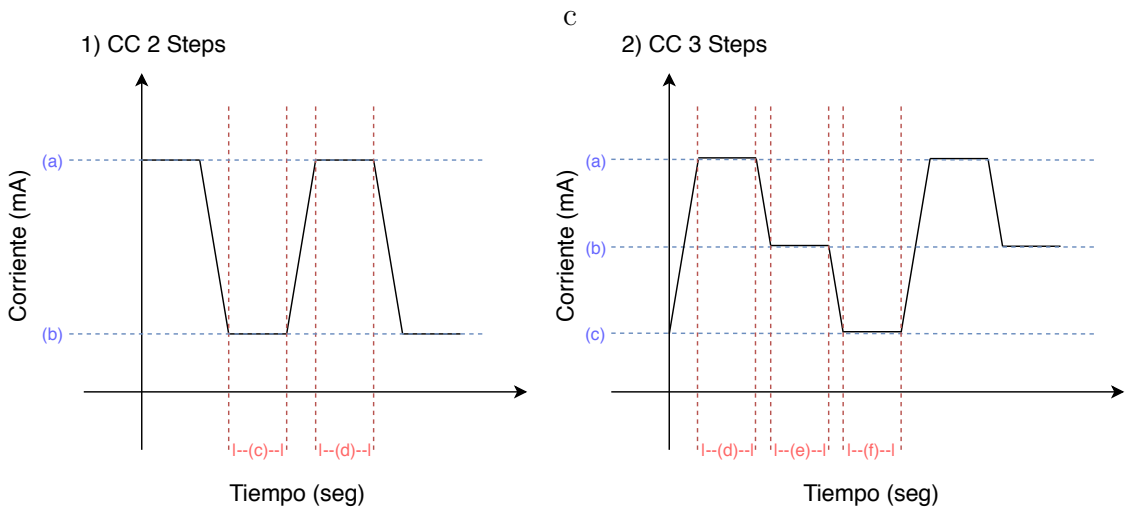

Figura 18: 1) CC 2 Steps: (a) Parámetro de corriente 1, (b) Parámetro de corriente 2 y (c) Parámetro de tiempo 1, (d) Parámetro de tiempo 2; 2) CC 3 Steps: (a) Parámetro de corriente 1, (b) Parámetro de corriente 2, (c) Parámetro de corriente 3 y (d) Parámetro de tiempo 1, (e) Parámetro de tiempo 2, (f) Parámetro de tiempo 3.

Tabla 3: Tamaño de los parámetros de corriente y tiempo para cada esquema de carga.

Método de carga	Corriente (mA):		Tiempo (seg):	
Esquema de carga	Min:	Max:	Min:	Max:
CC 2 Steps (1C)	0	1000	0	10
CC 3 Steps (1C)	0	1000	0	10
CC 2 Steps (2C)	0	2000	0	10
CC 3 Steps (2C)	0	2000	0	10

2. CC 3 Steps: Este esquema de carga percibe 3 parámetros de corriente y 2 parámetros de tiempo (Fig. 18 (2)).

La metodología de este método de carga se puede adaptar para cualquier tipo de batería y cualquier tamaño, en el caso particular de este trabajo se adapto para baterías de tipo LiPo de 1000 mA por lo que los limites para los parámetros de corriente son de 2C, esto significa que podemos elevar la corriente hasta el doble de capacidad de la batería, los parámetros específicos son los siguientes:

El método de carga solamente esta presente durante la etapa de CC de la batería hasta llegar a los 4.2 V, una vez alcanzado este voltaje el método de carga se detiene para permitir el proceso de estabilización los químicos internos de la batería, este

proceso tiende a tomar pocos segundos y el voltaje de la batería cae hasta los 4.1 V, en algunos casos cuando el proceso de carga toma muy poco tiempo el voltaje decae más llegando hasta los 4.07 V. Determinamos que estamos ante un problema de optimización, ya que es necesario encontrar los parámetros más eficientes para cargar un batería en un menor tiempo; existen combinaciones de parámetros que son mejores que otras y el objetivo es determinar mediante el uso de una técnica de optimización evolutiva las mejores combinaciones de parámetros.

6.3. Prototipo final

Para alcanzar las metas planteadas de forma satisfactoria, se hizo presente la necesidad de actualizar el prototipo inicial, específicamente en el dispositivo de carga. En base a los resultados obtenidos con el dispositivo de carga TP4056, se determinó que existe una mejor eficiencia en tiempo de carga con el método de carga propuesto en comparación con el método de carga convencional (Método de carga CC-CV), sin embargo, no fue posible implementar completamente el método de carga propuesto debido a las limitaciones del dispositivo. Entre estas limitaciones se encuentra el no poder cargar una batería a más de 1000 mA y, debido a la configuración del dispositivo, solamente se modifico el ciclo de carga durante una pequeña fracción de tiempo aproximadamente de 5 minutos.

Entre las principales características de este prototipo se encuentra el uso de una fuente programable por lo que se excluye el uso del dispositivo de carga TP4056, se adapta un sistema de descarga de baterías y se implementa una técnica de optimización que genera los parámetros de carga y los evalúa en base al tiempo del ciclo de carga. En el prototipo anterior era necesario conectar las 4 baterías descargadas para iniciar con el proceso de carga, después de finalizar con la ejecución del experimento se tenían que desconectar las 4 baterías y descargar manualmente y así, repetir nuevamente el procedimiento. Con estas tres actualizaciones el prototipo genera una completa autonomía por lo que ya no es necesario ningún tipo de intervención humana como en el caso del prototipo anterior. También es importante comentar que debido a ciertas situaciones solo se pudo conseguir una fuente programable por lo que el sistema solo es capaz de cargar una batería a la vez y no 4 como en el proto-

tipo anterior. Desde una perspectiva general el sistema se conforma de hardware y software, en la estructura del hardware podemos encontrar cuatro módulos:

1. Módulo de control
2. Módulo de carga
3. Módulo de descarga
4. Módulo de baterías

Para el software adaptamos un algoritmo genético que genera los parámetros de carga para la implementación del método de carga propuesto.

6.3.1. Estructura Física (Hardware)

La parte del hardware del sistema tiene diversos cambios a diferencia del prototipo inicial, estos cambio están presentes en el modulo de control y el nuevo modulo de descarga, a continuación se muestra detalladamente la composición y el funcionamiento de cada modulo.

6.3.1.1. Módulo de control

El módulo de control, como en el prototipo anterior, se encarga de controlar todo el sistema, activar y desactivar la carga de las baterías y monitorear el estado de las baterías. El monitoreo de las baterías consiste en revisar si la batería ya está completamente cargada, si se encuentra completamente descargada y analizar los niveles de voltaje cada 5 segundos. El sistema se compone de:

- Un ordenador en donde se aloja y computa el algoritmo evolutivo, este controla todo el sistema.
- Un Arduino que sensa los niveles de voltaje de la batería y controla un arreglo de relevadores.
- Un arreglo de relevadores que conecta la batería al sistema de carga y descarga.

6.3.1.2. Módulo de carga

El módulo de carga se compone de una fuente de poder programable y con comunicación serie bajo el protocolo Modbus RTU. La fuente de poder programable DPH5005 [Dps, 2019] es capaz de ofrecer hasta un voltaje máximo de 50 V y una corriente máxima de 5A, la precisión de la salida de voltaje y corriente es de ± (0.5 %) y cuando se encuentra a corriente constante o voltaje constante ofrece los valores de corriente y voltaje necesarios para mantener estas funciones.

6.3.1.3. Módulo de descarga

El módulo de descarga se compone de una bombilla que consume corriente de manera constante en base al voltaje de la batería.

6.3.1.4. Módulo de baterías

Este módulo se compone de una batería de 1000 mA, para cada experimento se utilizo una batería distinta (nueva) para evitar que el desgaste de la batería tenga una repercusión significativa en la eficiencia de los parámetros de carga.

6.3.2. Algoritmo genético (Software)

Se utilizó un algoritmo genético para generar los parámetros de carga de forma evolutiva, esto permitió parte de la autonomía del sistema puesto que ya no era necesario ingresar estos parámetros manualmente. Entre las principales características que tenemos al utilizar un algoritmo genético es que mientras se prueban los parámetros de carga, estos son evaluados mediante una función de aptitud en base al tiempo de carga y aparte, generamos datos importantes que pueden ser utilizados para generar en un trabajo futuro una función derivable y aplicar una técnica de inteligencia artificial basada en gradiente. La estructura del algoritmo genético se explica detalladamente a continuación.

6.3.2.1. Campo de búsqueda: Fenotipo

El campo de búsqueda del fenotipo se caracteriza por tener dos espacios de búsqueda, el primero es el espacio de la corriente y el otro en el espacio del tiempo. Estos a

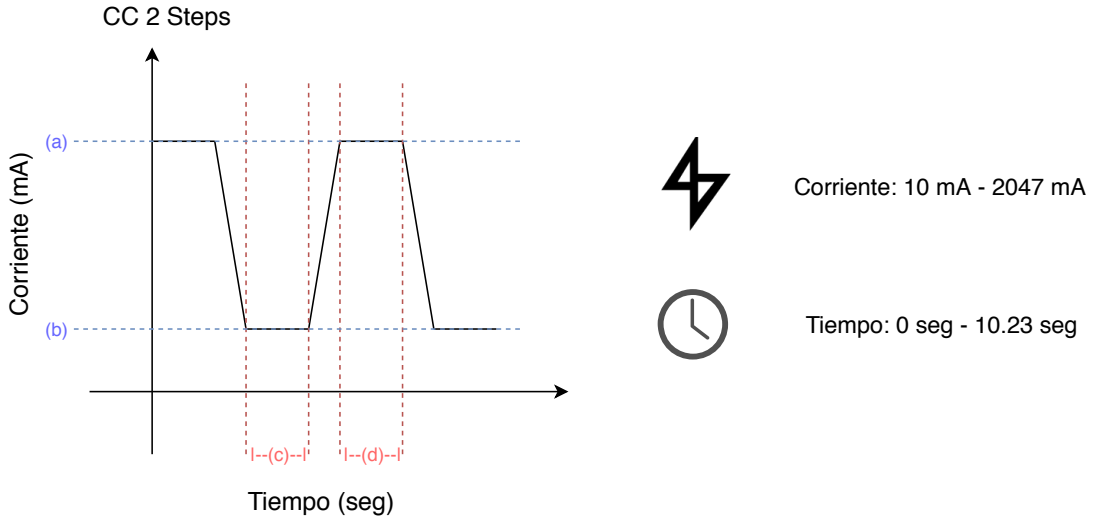

Figura 19: Representación del fenotipo.

su vez caracterizan los parámetros de carga que serán aplicados durante el proceso de carga; el espacio de la corriente va desde los l0 mA hasta los 2047 mA y en el caso del tiempo, va desde los 0 seg hasta los 10.23 seg (Fig. 19). Dependiendo del esquema de carga como se explico en la sección 4.2, se determinan 2 parámetros de corriente y 2 parámetros de tiempo para el esquema de carga CC 2 Steps y 3 parámetros de corriente y tiempo para el esquema CC3 Steps.

6.3.2.2. Campo de búsqueda: Genotipo

El campo de búsqueda del genotipo representa la estructura genética de los individuos en bits. La representación binaria de la corriente tiene un tamaño de 11 bits por cada parámetro de corriente, esto es en decimal de 0 mA - 2047 mA; en el caso del tiempo, su representación binaria es de 10 bits por lo que su valor decimal puede ir de 0 seg - 10.23 seg. El tamaño de esta estructura genética puede cambiar dependiendo del tiempo de esquema de carga, en el caso del esquema de carga CC 2 Steps es de 42 bits y de 63 bits para el esquema de carga CC 3 Steps (Fig. 20).

6.3.2.3. Población inicial

El algoritmo desarrollado ofrece la posibilidad de crear una población inicial de n individuos, sin embargo, se determinó que una población de 4 individuos era su-

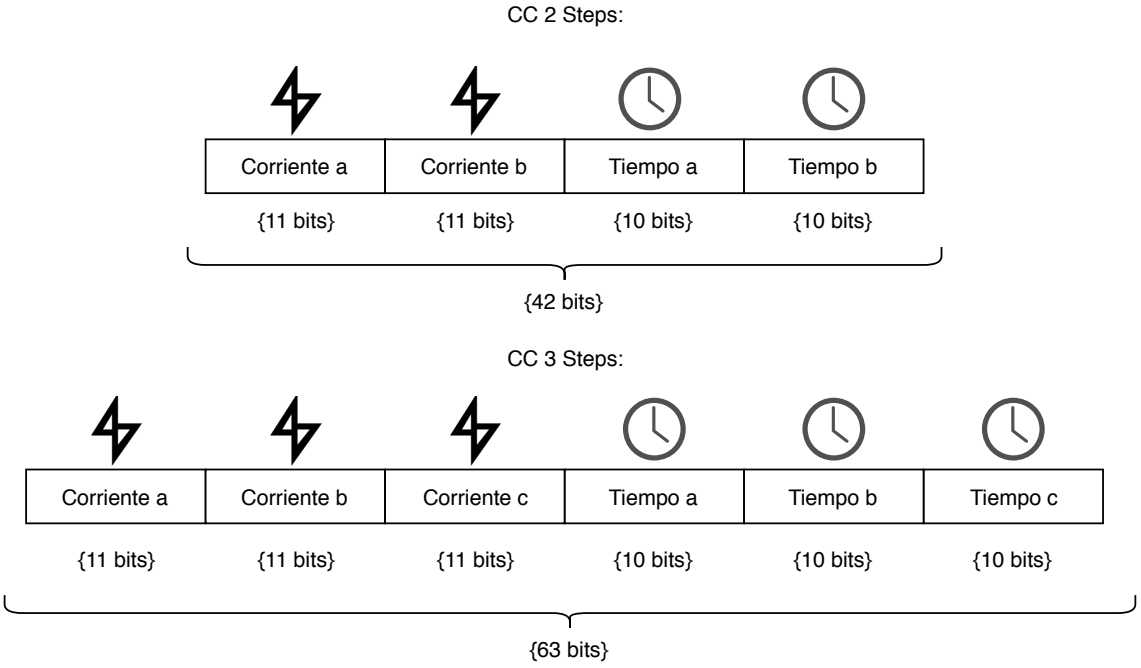

Figura 20: Representación del genotipo.

ficiente para conseguir resultados eficientes. Esto se debe principalmente a que el experimento no es una simulación, se trata de un experimento físico que utiliza baterías reales; diversos factores afectan la vida de la batería y por ende, la eficiencia del método de carga. El principal interés del experimento era comenzar con el proceso evolutivo y no tener una gran cantidad de individuos. Se inicia con una población de 4 individuos independientemente del esquema de carga y sus genes pueden generarse de dos formas:

- De forma aleatoria.
- Definidos por el usuario.

6.3.2.4. Función de Aptitud

Cada generación de individuos se evalúan de dos formas distintas, una que asigna un valor global al individuo y otro que asigna un valor local en base al rendimiento de la población.

1. Función de aptitud global: La evaluación del individuo se hace en base a todo el campo de solución.

Tabla 4: Valores de aptitud y su posición global en base a la función de aptitud global. Los valores se tomaron de la 5 generación en un experimento del esquema de carga CC 2 Steps.

Tiempos de carga (min)	Valor de aptitud	Posición global
A.- 38.35	A.- 360 - 38.35 = 321.65	1.- C
B.- 45.51	B.- 360 - 45.51 = 314.49	2.- D
C.- 28.24	C.- 360 - 28.24 = 331.76	3.- A
D.- 33.33	D.- 360 - 33.33 = 326.67	4.- B

2. Función de aptitud local: Se efectúa asignando un valor de aptitud al individuo en base al rendimiento de la población en una generación determinada.

La función de aptitud global evalúa cada individuo en base a una solución global se efectúa estableciendo un valor global, este valor global funciona como referencia para determinar que tan adaptado se encuentra el individuo. En la experimentación se propuso un valor de 360 y representa un tiempo de carga de 6 horas en el que asumimos que ese seria el peor de los casos, entonces el valor de aptitud global se obtiene haciendo una resta entre este valor y el tiempo de carga (Tabla 4).

Para el caso de la función de aptitud local el valor de aptitud que se asigna a cada individuo es en base al rendimiento de toda la población en una determinada generación, también es necesario conocer el contexto, el mejor individuo de la población puede no ser la mejor solución global, sin embargo en este caso estamos buscando el mejor individuo de la población. Para obtener este valor de aptitud es necesario promediar sumas los tiempos de carga de toda la población y después realizar una división entre ese valor local y el tiempo de carga por cada individuo (Tabla 5).

6.3.2.5. Selección

Se utilizó el método de muestreo estocástico universal (SUS; del inglés Stochastic Universal Sampling) el cual es similar a la técnica de la ruleta la diferencia es que en vez de utilizar un sólo punto fijo para la selección de un cromosoma se utilizan dos (Fig.21). La selección de los individuos es en base al tamaño que abarca en la ruleta de selección, el tamaño de cada individuo en la ruleta esta dado por su valor

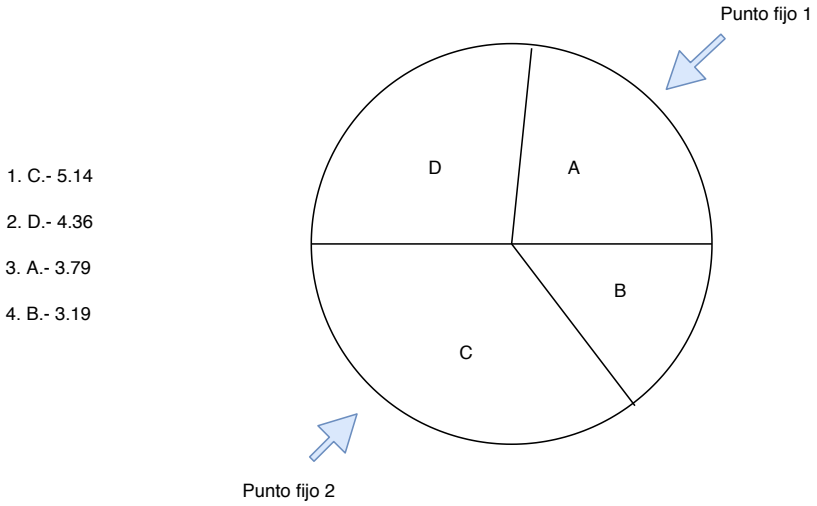

Figura 21: Ejemplo de selección por SUS.

de aptitud local, un mayor valor de aptitud local proveerá una mayor posibilidad de selección, sin embargo, no necesariamente un mayor espacio en la ruleta define la selección de este individuo como se observa en la figura 21, aunque los individuos más aptos son el C y el D, el método selecciono al individuo C y A.

6.3.2.6. Reproducción: Cruce

Se utilizaron diversos métodos de cruce, de hecho el algoritmo permite hacer hasta n cruces, sin embargo, en la teoría y experimentación se determinó que el cruce por dos puntos era el método más eficiente y se utiliza para cada parámetro de carga (Fig. 22).

Tabla 5: Valores de aptitud y su posición local en base a la función de aptitud local. Los valores se tomaron de la 5 generación en un experimento del esquema de carga CC 2 Steps.

Tiempos de carga (min)	Posición local
A.- 38.35	1. C.- 145.43 / 28.24 = 5.14
B.- 45.51	2. D.- 145.43 / 33.33 = 4.36
C.- 28.24	3. A.- 145.43 / 38.35 = 3.79
D.- 33.33	4. B.- 145.43 / 45.51 = 3.19
Total: 145.43	Total: 16.48

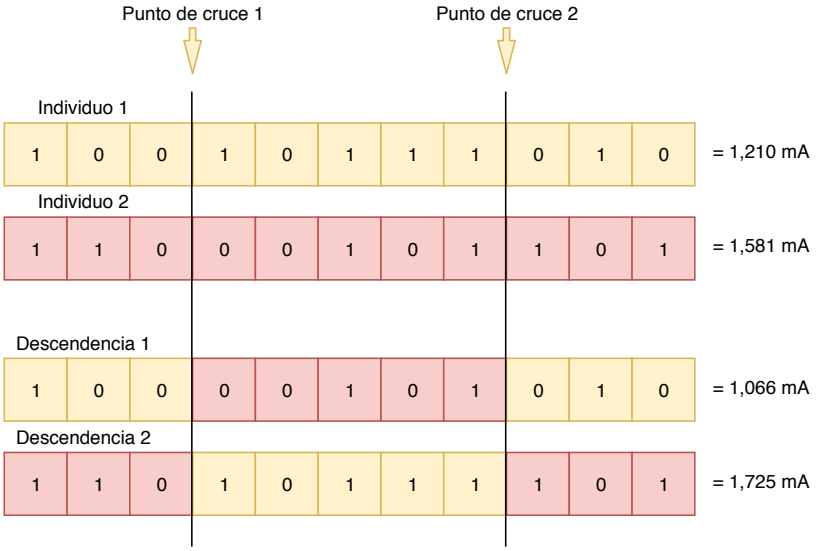

Figura 22: Ejemplo de cruce por 2 puntos.

6.3.2.7. Reproducción: Mutación El proceso de mutación en este caso especifico se dejo en solamente un bit de cada parámetro de carga, esto significa que 4 bits del genotipo para el esquema de carga CC 2 Steps pueden mutar y 6 para el genotipo del esquema de carga CC 3 Steps. El parámetro de mutación se inicializa en 0.05 o bien un 5 % de probabilidad de mutación (Fig.23).

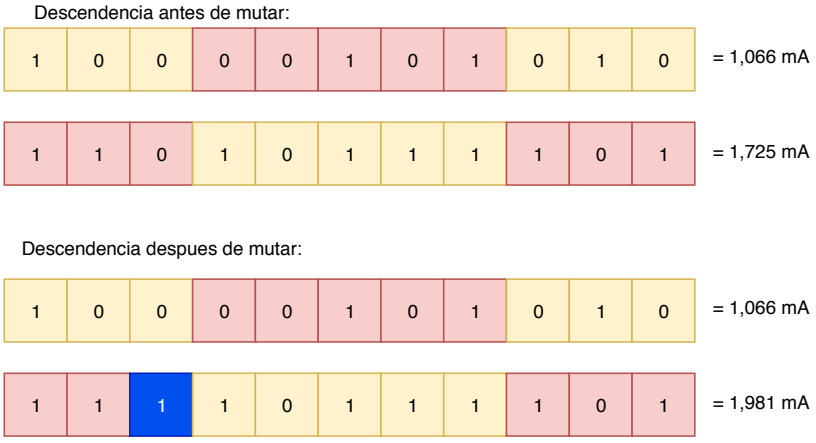

Figura 23: Ejemplo de mutación en un bit.

7. Resultados

Los resultados que se obtuvieron durante el proceso de experimentación apuntaron a diversas conclusiones que serán expuestas de manera objetiva. Es necesario explicar detalladamente el proceso de experimentación para la justificar los resultados, este proceso se dividió principalmente en dos partes: primer prototipo y prototipo final.

7.1. Resultados prototipo inicial

El desarrollo y arquitectura de este prototipo se expuso en la sección anterior, el prototipo solamente contaba con un modulo de control, modulo de carga y el modulo de baterías. Por esta situación no fue posible adaptar un algoritmo evolutivo que generara los parámetros de carga de forma automática como se hizo después, el experimento fue totalmente empírico con el fin de comprobar si era posible mejorar el proceso de carga de una batería modificando los parámetros de carga de un método especifico. Se utilizaron dos métodos de carga en este experimento, el método de CC-CV y el método de carga por pulsos, esto con el fin de crear un punto de comparación entre los dos métodos de carga. Para el proceso de carga con el método de carga CC-CV se conectó la batería al circuito de carga y este operó de manera típica sin ningún cambio externo. Se hicieron 5 ejecuciones con 4 baterías distintas, eso dio como resultado 20 ejecuciones de las cuales se tomo el mejor de los casos y el peor de los casos con los resultados en la tabla 6.

En el segundo de los casos, se presento una situación en la que por cuestiones del dispositivo TP4056, se limitaba la implementación del método de carga por pulsos

Tabla 6: Mejor y peor de los casos con el método de carga CC-CV.

Batería	Método de carga	Mejor de los casos (min)	Peor de los casos (min)
1	CC-CV	133	137
2	CC-CV	132	136
3	CC-CV	133	134
4	CC-CV	135	136

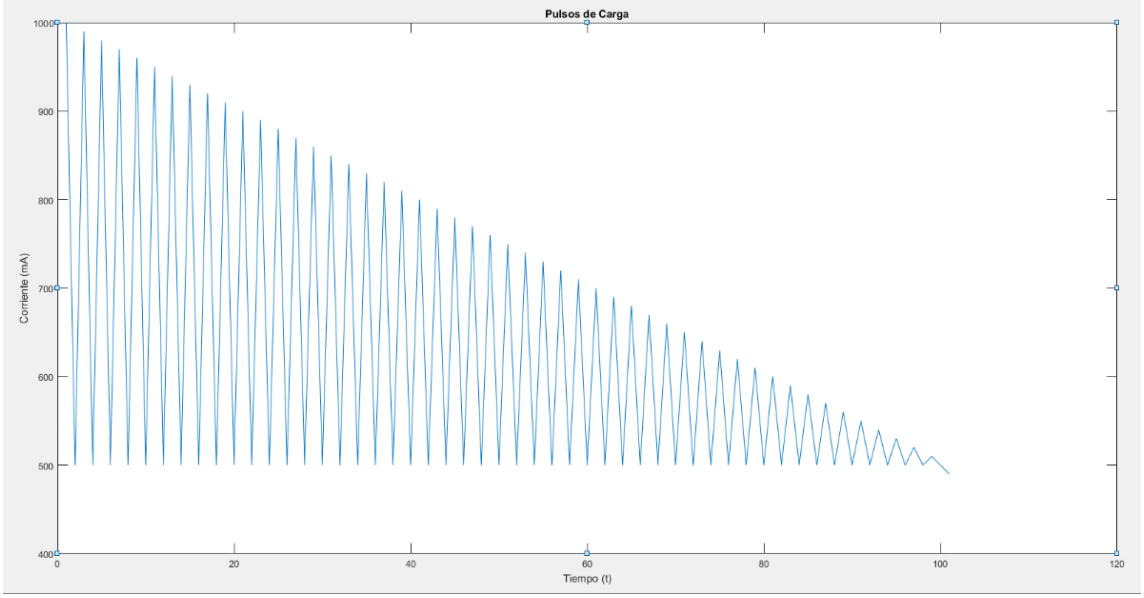

Figura 24: Método por pulsos aplicado en el prototipo inicial.

a solamente modificar una parte parcial del proceso de carga, era una pequeña fracción de tiempo, sin embargo se logro una diferencia importante en el tiempo de carga, aunque el tiempo de carga que se redujo fue relativamente pequeño, se logro determinar que modificando los parámetros en un método de carga determinado efectivamente reduce el tiempo del ciclo de carga. El lapso de tiempo en el proceso de carga donde se implemento el método de carga por pulsos fue de unos minutos, aproximadamente 1 a 2 minutos. En ese lapso de tiempo la corriente de salida oscilaba entre los valores de 1000 mA y 500 mA (Fig. 24). De igual manera se hicieron 5 ejecuciones en las cuatro baterías, de las cuales se tomo el mejor y el peor de los casos como se muestra en la tabla 7.

Si observa una diferencia en los tiempos de carga cuando se utiliza el dispositi-

Tabla 7: Mejor y peor de los casos con el método de carga por pulsos.

Batería	Método de carga	Mejor de los casos (min)	Peor de los casos (min)
1	CC-CV	124	125
2	CC-CV	125	126
3	CC-CV	124	127
4	CC-CV	125	127

vo TP4056 en su operacion normal y aplicando pulsos en una fracción de tiempo logrando concluir que realmente los parámetros de carga tienen un impacto directo en el tiempo de carga. Estos resultados dieron inca pie a implementar un nuevo prototipo que no limitara la implementación de este método de carga.

7.2. Resultados prototipo final

Los resultados presentados en la sección anterior permitieron el desarrollo de un prototipo que no estuviera tan limitado. En el caso del prototipo anterior se utilizo un dispositivo de carga TP4056, el cual, permitía modificar una pequeña fracción de tiempo en el ciclo de carga. La nueva implementación contempla el uso de una fuente de poder programable, en donde ahora seria posible modificar todo el ciclo de carga de una batería. Además de esto, se implemento un algoritmo genético que se encargaba de generar los parámetros de carga a los que las baterías tendrían que someterse de forma automática. El sistema permitió una completa autonomía del sistema por lo que el tiempo de experimentación fue sencillo de ejecutar, solamente dejarlo ejecutándose durante una determinada cantidad de tiempo. En total se ejecutaron 4 experimentos con las características generales mostradas en la tabla 8. La duración total del experimento completo fue de 753.63 horas o 31.40 días, este tiempo contempla el tiempo de carga, el tiempo de descarga y el tiempo de reposo entre cada proceso respectivamente que es de 10 minutos. En la tabla 8 se muestra detalladamente los resultados de cada esquema de carga. En cada uno de los experimentos se tomaron datos importantes como el tiempo de carga de toda la población determinar su rendimiento, de igual forma se tomo el mejor de los casos en cada población para determinar como iba mejorando a través de las generaciones.

Tabla 8: Duración del experimento en cada esquema de carga.

Esquema de carga	2 Steps 1C	2 Steps 2C	3 Steps 1C	3 Steps 2C
Tiempo total de ejecución (hrs)	241.48	200.10	175.35	136.70

7.2.1. CC 2 Steps a 1C

Los resultados de este experimento fueron bastante satisfactorios, no solamente por que se logro completar la hipótesis de la tesis, sino por que también se redujo el tiempo de carga de una manera bastante considerable. Para contextualizar un poco se detallara brevemente la configuración que se utilizo y que ya fue explicada en el capitulo anterior.
Se utilizo el método de carga propuesto en su esquema de carga CC 2 Steps, por lo que se utilizaron 2 parámetros de corriente y 2 parámetros de tiempo. En el caso del algoritmo genético, la configuración del experimento se muestra en la tabla 9.

Se ejecutaron un total de 55 generaciones lo que llevo a un total de 241.48 horas ininterrumpidas. El rendimiento de la población junto con su progreso conforme a las generaciones se puede ver en la figura 25, el comportamiento que se observa en los resultados es fácil de interpretar.

Se inicio con estructura genética aleatoria en los cromosomas, el rendimiento de la primer generación no fue muy bueno dando como resultado 429.21 minutos, sin embargo, el mejor individuo en esa generación obtuvo un tiempo de carga de 43.82 minutos, que en comparación con los 132 minutos del primer prototipo ya se contaba con un mejor resultado. En la segunda generación se observa un resultado peor que el primero, dando un tiempo de carga de 926.18 minutos, esto pudo deberse a que el algoritmo busco en otro espacio de solución donde no obtuvo un resultado favorable, a partir de esa generación el comportamiento de toda la población fue bastante estable y con tendencia a disminuir su tiempo de carga como se observa

Tabla 9: Configuración del algoritmo genético en experimento CC 2 Steps a 1C.

Configuración	Valor
Tamaño cromosoma	42 bits
Elitismo	No
Estructura genética	Aleatoria
Método de selección	SUS
Método de cruce	2 puntos
Mutación	0.05 %

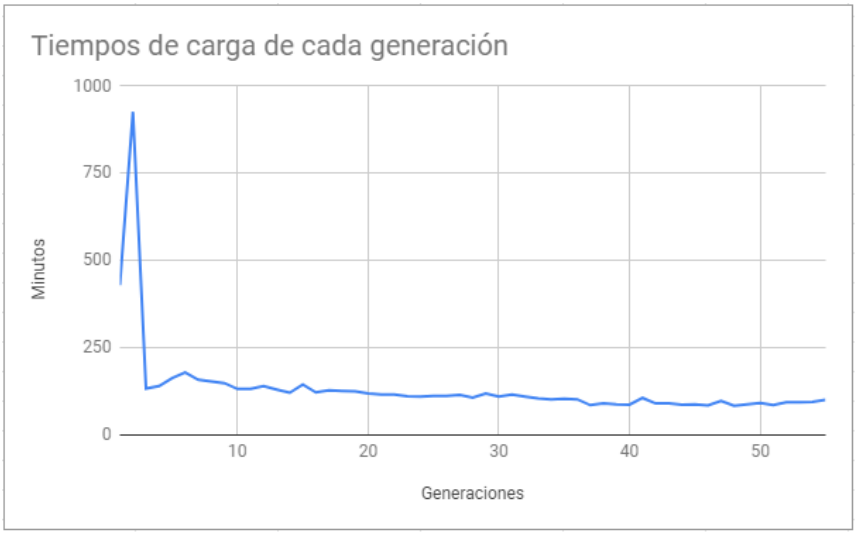

Figura 25: Rendimiento de la población en relación con las generaciones en experimento CC 2 Steps a 1C.

en la figura 25. El mejor tiempo que obtuvo la población específicamente de este experimento se registro en la generación 48 con un resultado de 83.97 min, a partir de esa generación ya no hubo una mejora en el tiempo de carga de la población.
En el caso del mejor individuo a través de las generaciones, los resultados pueden verse en la figura 26.

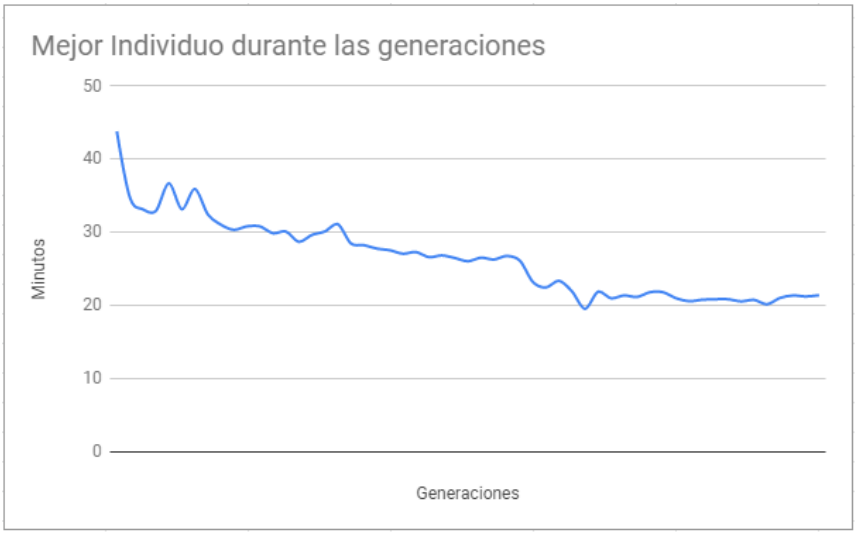

Figura 26: Rendimiento del mejor individuo en relación con las generaciones en experimento CC 2 Steps a 1C.

En este caso se logra observar de una manera mas clara una tendencia a disminuir el tiempo de carga, los datos resumidos se observan en la tabla 10.

Tabla 10: Información estadística del experimento CC 2 Steps a 1C.

Información	Individual (min)	Población (min)
Mejor de los casos	19.55	83.97
Peor de los casos	631.39	926.18
Media	33.41	133.51
Desviación estándar	44.67	119.02

También se realizo una comparación entre los tiempos de carga y las tiempos de descarga, esto debido a que se observo un comportamiento bastante peculiar en algunos ciclos de carga. Cuando el ciclo de carga tenia una mayor duración en el tiempo de carga, el tiempo de descarga era mayor y cuando la duración del tiempo de carga era menor la descarga finalizaba en menor tiempo, visto desde una perspectiva general tenemos la figura 27.

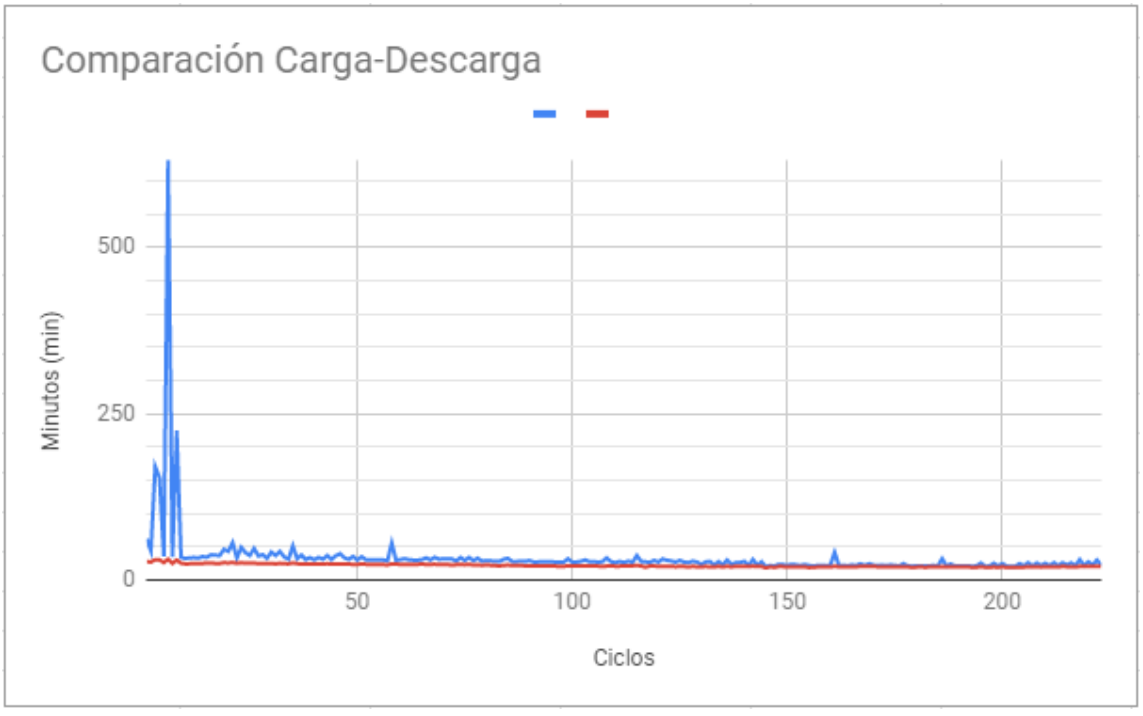

Figura 27: Comparación carga-descarga del experimento CC 2 Steps a 1C.

En ella podemos observar todos los tiempos de los ciclos de carga con su respectivo tiempo de descarga, si bien esta es una gráfica que adjunta todos los ciclos del experimento, durante ciertos ciclos podemos observar como la linea de descarga cambia su comportamiento. Por ejemplo si hacemos un comparativo entre los 50 y

150 ciclos se logra observar a mayor detalle como en los ciclos en donde el tiempo de carga es mayor, el tiempo de descarga aumenta ligeramente (Fig. 28). Con estos resultado podemos concluir que el proceso químico al que se someten los materiales internos de una batería tienen mayor eficiencia en tiempo de descarga (o retiene mas energía) si el proceso de carga es mas lento. Que un proceso de carga sea mas lento indica el uso de parámetros de corriente moderados en comparación con la capacidad de carga de la batería, o bien, utilizando un parámetro de carga grande y otro mas pequeño, como lo hace el método de carga propuesto.

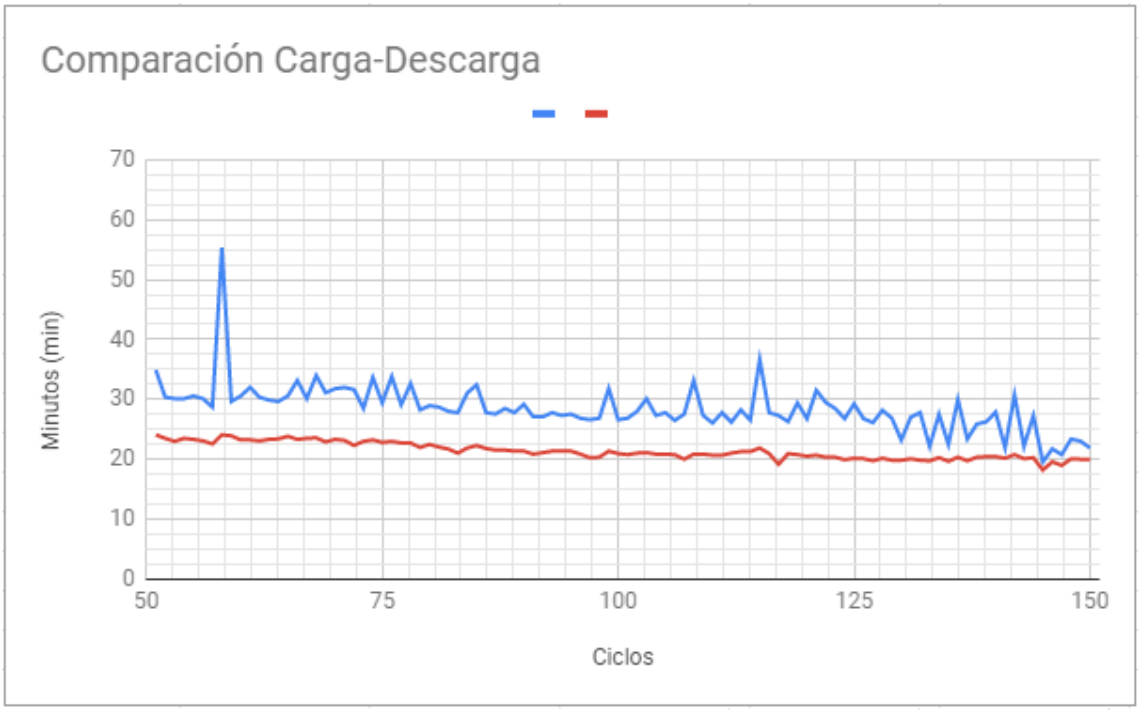

Figura 28: Comparación carga-descarga del experimento CC 2 Steps a 1C del ciclo 50 al 150.

7.2.2. CC 3 Steps a 1C

El método de carga propuesto con la variante CC 3 Steps utiliza 3 parámetros de corriente y 3 parámetros de tiempo a una corriente máxima de 1C, por lo que los parámetros de corriente solamente pueden llegar a 1024 mA, la configuración del algoritmo genético se observa en la tabla 11. La estructura genética de cada individuo se genero de forma aleatoria y como se explica en la sección anterior, tienen un tamaño de 63 bits.

En total se ejecutaron 35 generaciones lo que llevo un tiempo total de 175.35 horas

Tabla 11: Configuración del algoritmo genético en experimento CC 3 Steps a 1C.

Configuración	Valor
Tamaño cromosoma	63 bits
Elitismo	No
Estructura genética	Aleatoria
Método de selección	SUS
Método de cruce	2 puntos
Mutación	0.05 %

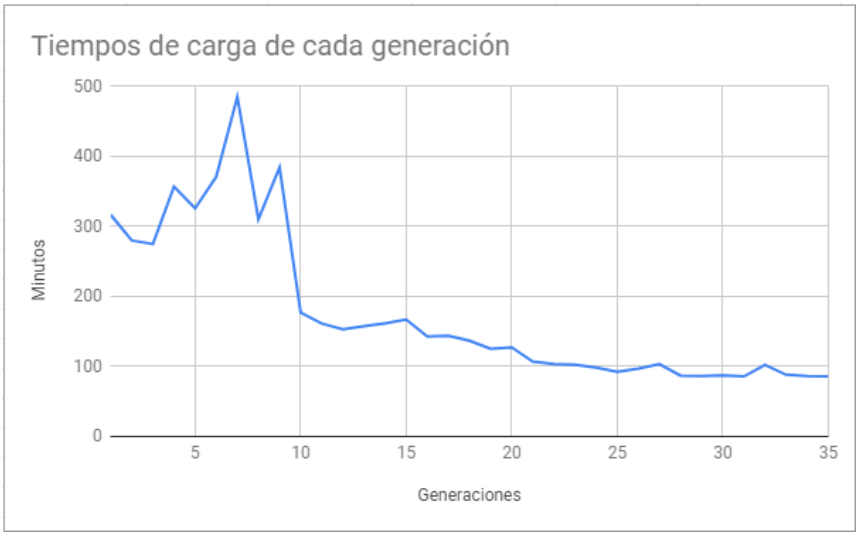

Figura 29: Rendimiento de la población en relación con las generaciones en experimento CC 3 Steps a 1C.

ininterrumpidas, dentro de este cálculo se contempla el tiempo de carga, tiempo de descarga y el tiempo que se toma para que la batería se estabilice después de cada ciclo. La evaluación de cada generación con respecto al tiempo se puede observar en la figura

El comportamiento del rendimiento de la población en base a cada generación es muy parecida al experimento anterior, el CC 2 Steps a 1C. Una de las principales diferencias es que esta implementación del método de carga propuesto tardo un poco mas en encontrar resultados óptimos, en el primer experimento la tendencia a mejorar los tiempos de carga inicio desde la tercera generación, en este caso fue a partir de la décima generación. En la figura 29 podemos observar que a partir de la tercera

generación que se presenta un comportamiento bastante errático sin una tendencia fija a mejorar o empeorar la eficiencia de carga, esto se debe a que el algoritmo por iniciar en condiciones aleatorias posiciona cada individuo en puntos distintos dentro del espacio de solución, estos a su vez requieren compartir información genética para comenzar a mejorar a partir del mas adaptado. A partir de la novena generación, se observa claramente una tendencia a mejorar el tiempo de carga a nivel población, lo que indica que se encontró un mínimo local. El comportamiento del mejor individuo de cada generación en relación con el tiempo de carga se puede observar en la figura 30.

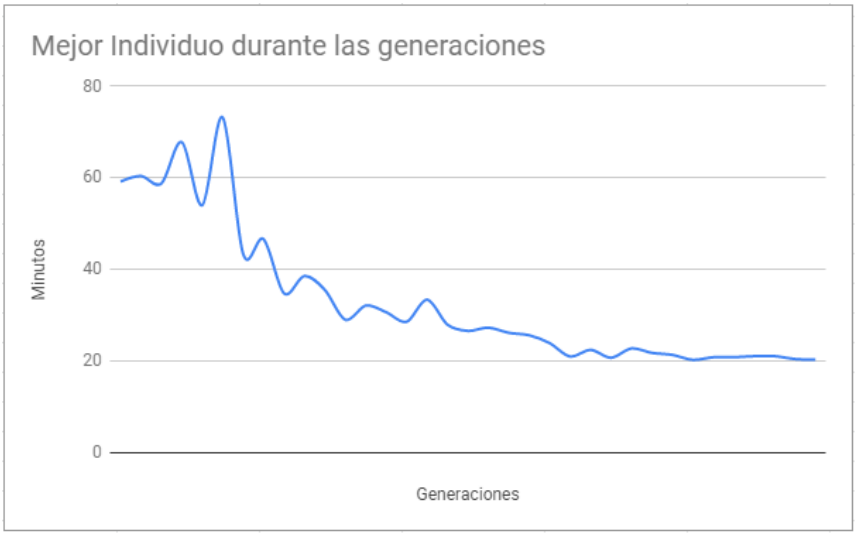

Figura 30: Rendimiento del mejor individuo en relación con las generaciones en experimento CC 3 Steps a 1C.

Este caso también presenta claramente una tendencia a disminuir el tiempo de carga, sin embargo, se puede observar que después de que se obtuvo un mejor resultado en comparación con la generación anterior, el tiempo de carga del mejor individuo de la siguiente generación empeora, tal vez no significativamente pero si existe un cierto nivel de variación. Los principales resultados del experimento se pueden observar en la tabla 12, la diferencia entre el mejor de los casos del experimento anterior y este es de casi 1 minutos beneficiando al experimento anterior, no es mucha diferencia si se contempla que en este caso particular de la implementación del método de carga propuesto se puede tener 3 parámetros de corriente y 3 parámetros de tiempo. Esto por consecuencia da un espacio de solución mas grande que en la implementación de CC 2 Steps.

Tabla 12: Información estadística del experimento CC 3 Steps a 1C.

Información	Individual (min)	Población (min)
Mejor de los casos	20.45	83.97
Peor de los casos	161.67	484.98
Media	43.72	176.21
Desviación estándar	33.12	108.72

En los resultados expuestos de carga y descarga (Fig. 31) se logra observar como los parámetros de carga fueron evolucionando y adaptándose de tal manera que al final del experimento la eficiencia del tiempo de descarga de la batería se iguala al tiempo de carga, esto nos indica principalmente que el tiempo de carga es completamente dependiente de dos factores principales, los parámetros de carga y la capacidad de retención de energía de la batería.

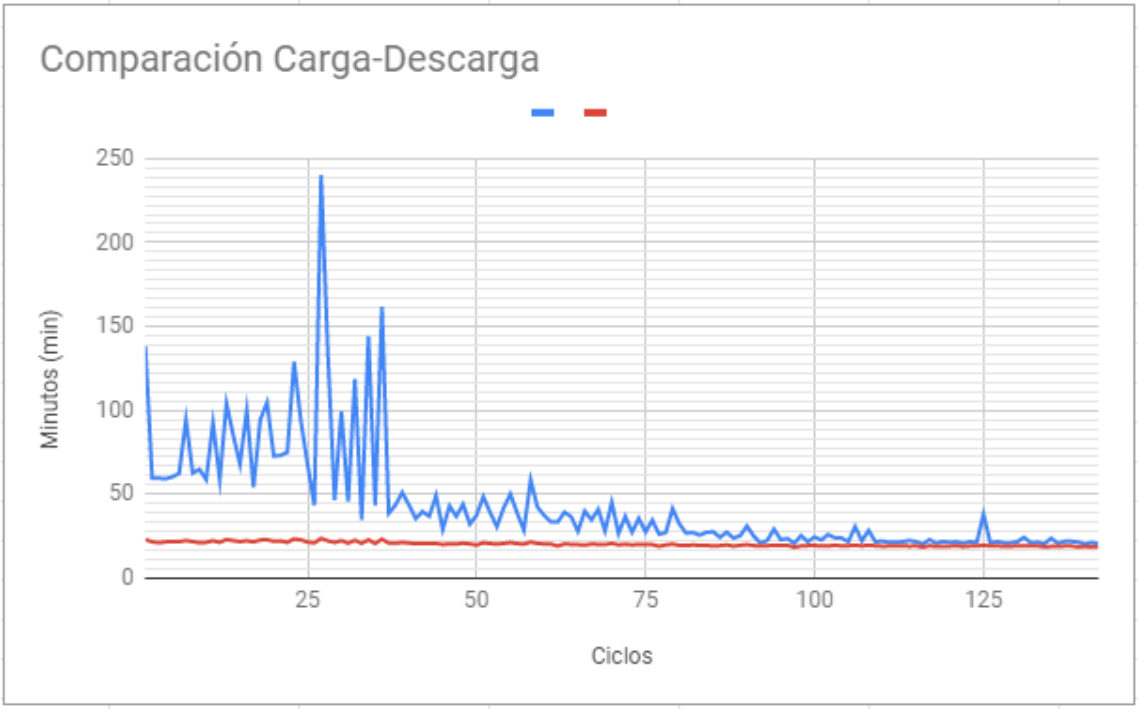

Figura 31: Comparación carga-descarga del experimento CC 3 Steps a 1C.

7.2.3. CC 2 Steps a 2C

A diferencia de los resultados expuestos anteriormente derivados de la experimentación del método de carga propuesto a 1C, los presentes resultados se derivan

de un experimento elevando la carga a 2C. Esto significa que en los parámetros de corriente el limite de corriente máxima se eleva de los 1000 mA a 2000 mA, esto de manera empírica se puede concluir que debería existir una diferencia en el tiempo de carga en comparación con los experimentos a 1C, debido a que si existe una mayor corriente las baterías deberían cargarse mas rápido, sin mas preámbulo se presentan las principales características del experimento y sus resultados.

El experimento se basa en el método de carga propuesto CC 2 Steps por lo que se utilizan dos parámetros de carga y dos parámetros de tiempo, sin embargo, los parámetros de carga como ya se mencionó anteriormente, son a 2C. La configuración del algoritmo genético puede observarse en la tabla 13.

Tabla 13: Configuración del algoritmo genético en experimento CC 2 Steps a 2C.

Configuración	Valor
Tamaño cromosoma	42 bits
Elitismo	No
Estructura genética	Aleatoria
Método de selección	SUS
Método de cruce	2 puntos
Mutación	0.05 %

El experimento tuvo un total de 74 generaciones, este fue el experimento con un mayor numero de generaciones en comparación con los demás con un tiempo total de 200 horas interrumpidas y la razón puede observarse claramente en los resultados a nivel población y a nivel individual. Los resultados a nivel población pueden observarse en la Fig. 32, se observa claramente que no existe una tendencia fija, en un momento se observa que el tiempo de carga a nivel población va mejorando, sin embargo al cabo de una o dos generaciones este empeora, este comportamiento sigue presentándose durante todo el experimento. Debido a este comportamiento, la condición de paro del algoritmo no se efectuó correctamente, debido a que el algoritmo mejoraba al cabo de una o dos generaciones y volvía a empeorar de la misma manera, así sucesivamente hasta lograr burlar la condición de paro.

La explicación que podría darse a este comportamiento es que, debido a que ahora nos encontramos con un campo de búsqueda mas grande debido a que el parámetro

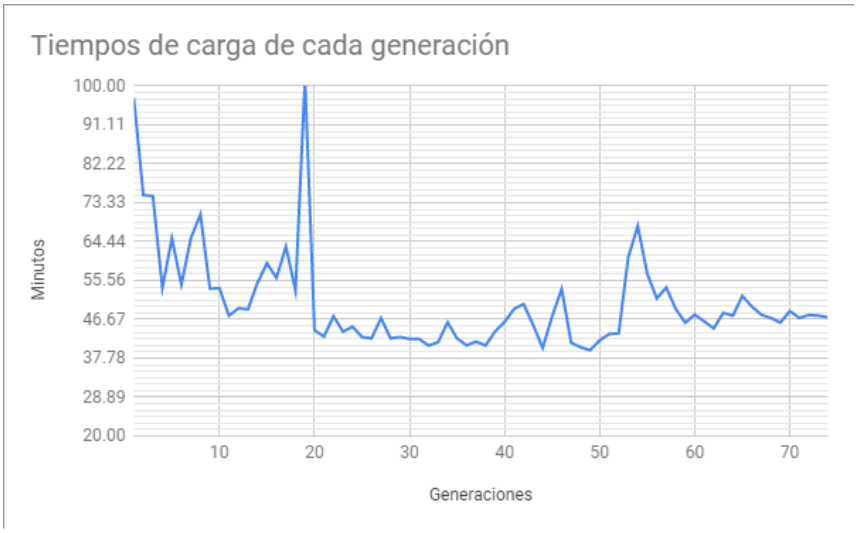

Figura 32: Rendimiento de la población en relación con las generaciones en experimento CC 2 Steps a 2C.

de carga va desde 0 mA a 2000 mA y no de 0 mA a 1000 mA como se manejaba anteriormente, el algoritmo cambia rápidamente de un mínimo local a otro transformando la búsqueda evolutiva en una búsqueda mas probabilística, posiblemente la configuración del algoritmo genético no fue la mas eficiente, sin embargo, se hicieron distintos cambios al experimento, como el uso de elitismo que concluyo con el mismo resultado. Este comportamiento probabilístico se puede observar de una forma mas clara en los resultados a nivel individuo del experimento (Fig. 33).

Estos resultados nos dan la apertura de entender que no siempre un algoritmo genético puede otorgar un resultado eficiente, definitivamente se pueden realizar cambios al algoritmo y adaptarlo para un entorno con un mayor campo de búsqueda. A pesar de todos estos resultados, se logro mejorar en un 45.5 % el tiempo de carga en comparación con el mejor resultado de los experimentos a 1C, sin embargo, no se puede atribuir esta mejora completamente al algoritmo genético ya que el aumento de la corriente influyo directamente en la disminución del tiempo de carga de las baterías.

Los resultados del experimento se observan el tabla 14, obteniendo un tiempo de carga de 8.91 minutos para el mejor de los casos y 39.62 minutos para el peor de los casos.

A diferencia de los experimentos anteriores en donde la linea de los tiempos de

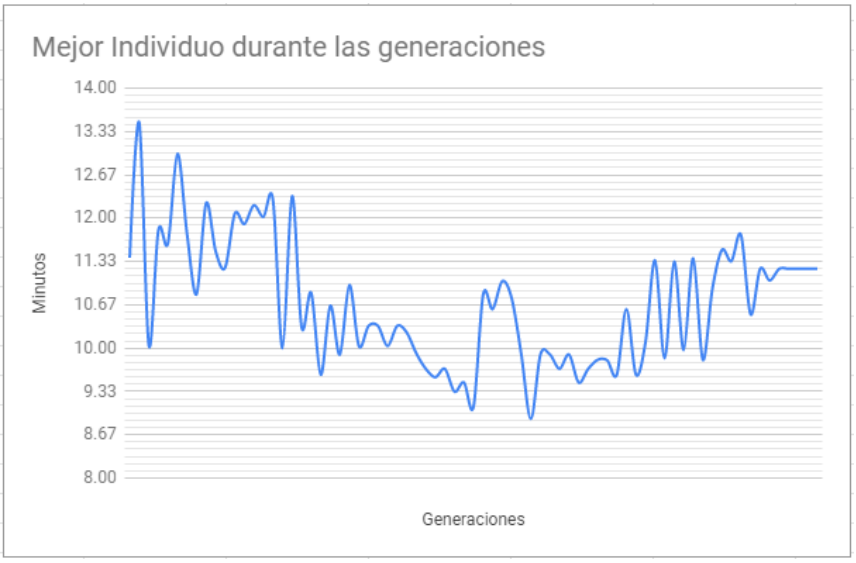

Figura 33: Rendimiento del mejor individuo en relación con las generaciones en experimento CC 2 Steps a 2C.

Tabla 14: Información estadística del experimento CC 2 Steps a 2C.

Información	Individual (min)	Población (min)
Mejor de los casos	8.91	39.62
Peor de los casos	37.79	101.58
Media	12.6	49.9
Desviación estándar	4.7	11.51

carga estaba sobre la linea de los tiempos de descarga, aquí podemos observar el efecto contrario, la linea de los tiempos de carga esta por debajo de los tiempos de descarga (Fig. 34).

7.2.4. CC 3 Steps a 2C

Este experimento utiliza el método de carga propuesto CC 3 steps con la configuración de carga a 2C. A diferencia del experimento anterior, en este si se puede notar con un poco mas de claridad la tendencia a mejorar o empeorar el tiempo de carga de la batería. La configuración del algoritmo se puede ver en la tabla 15.

El experimento finalizo con un total de 45 generaciones y una duración de 136 horas interrumpidas. Como se hace mención en el inicio de este experimento, se pue-

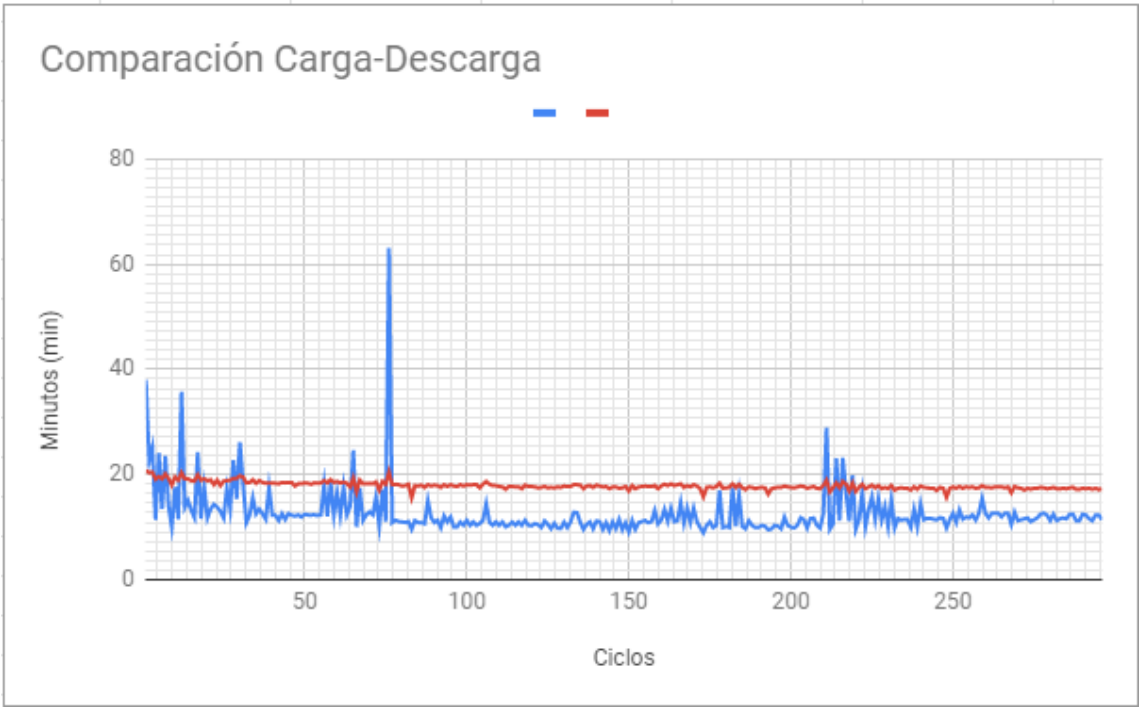

Figura 34: Comparación carga-descarga del experimento CC 2 Steps a 2C.

de observar con mayor facilidad la tendencia del comportamiento de los resultados, en la figura 35. A partir de la segunda generación, se observa que el resultado en los tiempos de carga a nivel población mejora drásticamente pasando de los 129 minutos a los 54 minutos en una sola generación, a partir de ahí, el algoritmo evoluciona de una forma precisa y mantiene una tendencia fija. Es a partir de la generación 35 es en donde el algoritmo comienza a generar una tendencia negativa en los resultados. A diferencia del comportamiento del experimento anterior, estos resultados indican que se puede ampliar el campo de búsqueda de un algoritmo genético y mantener

Tabla 15: Configuración del algoritmo genético en experimento CC 3 Steps a 2C.

Configuración	Valor
Tamaño cromosoma	63 bits
Elitismo	No
Estructura genética	Aleatoria
Método de selección	SUS
Método de cruce	2 puntos
Mutación	0.05 %

Figura 35: Rendimiento de la población en relación con las generaciones en experimento CC 3 Steps a 2C.

resultados eficientes si se tiene un mayor numero de parámetros, por ejemplo, en el experimento anterior solamente teníamos 2 parámetros de corriente y 2 parámetros de tiempo, en este experimento contamos con 3 parámetros de corriente y 3 parámetros de tiempo.

Pasando a los resultados individuales del mejor individuo por generación (Fig. 36), podemos observar un comportamiento con una tendencia un poco variable, sin embargo se logra observar como el algoritmo encuentra mínimos locales en el campo de búsqueda y después se salta hacia otro, en algunas de las ocasiones este no es un mínimo local muy bueno, sin embargo, en otros casos si se encuentra con uno que logra disminuir bastante el tiempo de carga de la batería. El mejor de los casos logro cargar una batería en 10.44 minutos, los resultados se presentan en la tabla 16.

Tabla 16: Información estadística del experimento CC 3 Steps a 2C.

Información	Individual (min)	Población (min)
Mejor de los casos	10.44	45.51
Peor de los casos	44.78	133.44
Media	13.60	54.21
Desviación estándar	4.35	13.58

La comparación de los tiempos de carga y descarga nos indica los mismos resul-

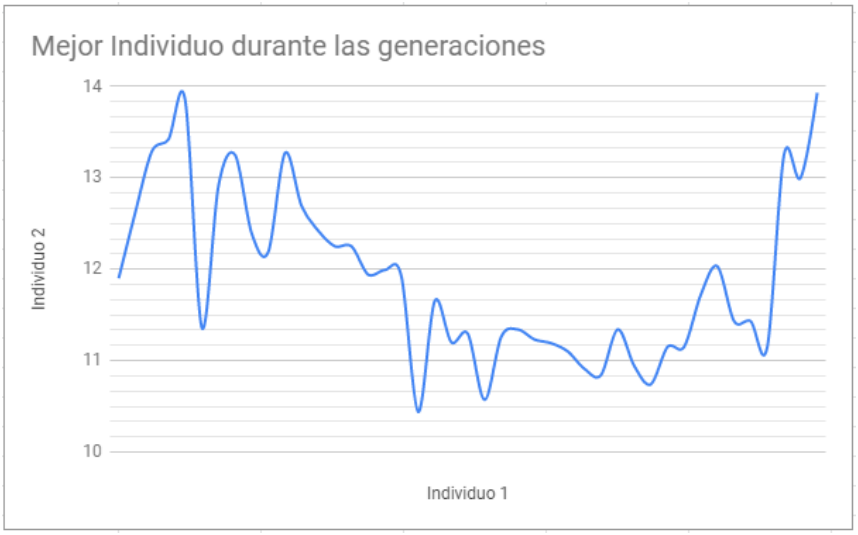

Figura 36: Rendimiento del mejor individuo en relación con las generaciones en experimento CC 3 Steps a 2C.

tados que el experimento anterior, los tiempos de descarga fueron mas altos que los tiempos de descarga (Fig. 37).

Los resultados al igual que los resultados a nivel población y individual fueron variables y con una tendencia fija pero con un cierto grado de variación. Sin embargo, se logra observar con mayor claridad el comportamiento de la batería en su capacidad de retención de energía, se observa que en los picos en donde existe un mayor tiempo de carga, el tiempo de descarga es ligeramente mayor.

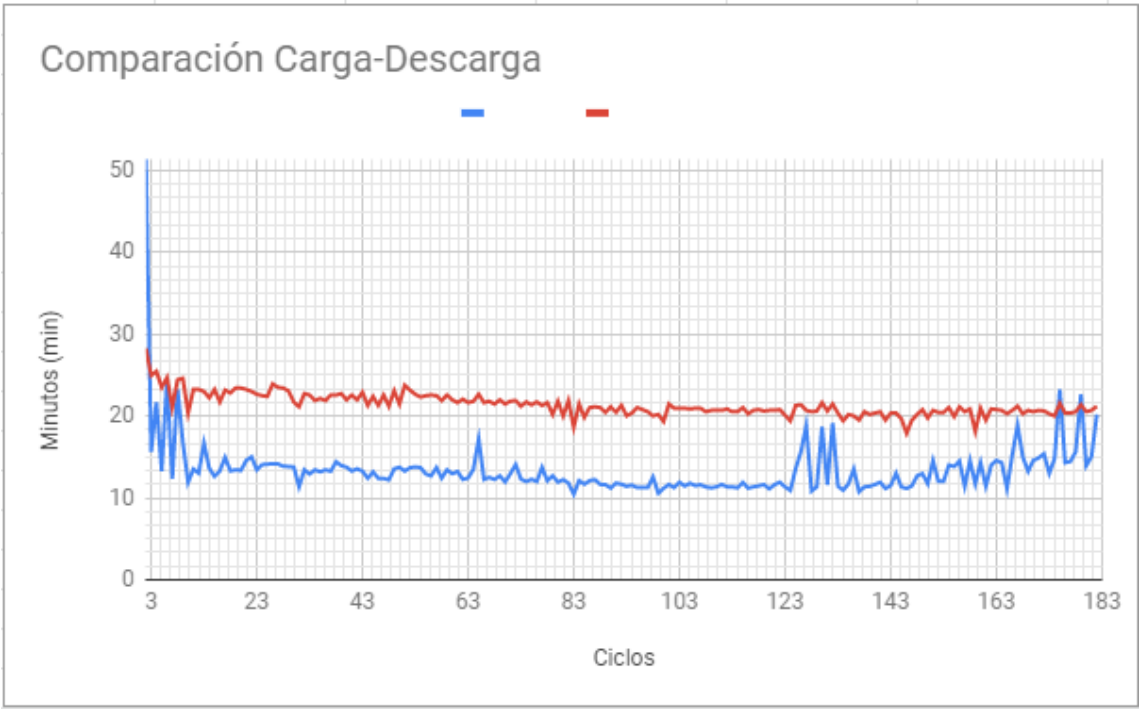

Figura 37: Comparación carga-descarga del experimento CC 3 Steps a 2C.

7.2.5. Comparación de la experimentación final

Los resultados expuestos en la sección anterior manifiestan que el uso de algoritmos genéticos y una buena estructura de hardware facilita la búsqueda e implementación de metodologías que mejoren la eficiencia del tiempo de carga en baterías, en este caso particular de baterías de tipo LiPo. Los resultados obtenidos de estos experimentos permiten un mayor entendimiento del funcionamiento de las baterías, por ejemplo, su tiempo de carga, tiempo de descarga, retención de energía y el efecto de una carga rápida en una batería. En este ultimo podemos concluir en base a los resultados expuestos en la tabla 17, que el efecto de una carga rápida influye directamente en la cantidad de energía retenida por la batería, esto evidentemente se debe al proceso químico que se lleva a cabo dentro de la batería.

Propiedad	CC 2 Steps 1C	CC 2 Steps 2C	CC 3 Steps 1C	CC 3 Steps 2C
Generaciones	55	74	35	45
Tiempo de ejecución	241.48	200	175.35	136
Tiempos de carga (min)				
Mejor de los casos	19.55	8.91	20.45	10.44
Peor de los casos	631.39	37.79	161.67	44.78
Media	33.41	12.6	43.72	13.60
Desviación estándar	44.67	4.7	33.12	4.35
Tiempo de descarga (min)				
Mejor de los casos	30.39	20.78	23.24	28.26
Peor de los casos	18.22	16.52	18.39	18.14
Media	21.66	17.88	20.29	21.40
Desviación estándar	2.40	0.71	1.23	1.39

8. Conclusión

Definitivamente es posible determinar mediante el uso de algoritmos genéticos parámetros de carga con los cuales una batería se carga más rápido, aplicándolos de forma sistemática mediante la implementación de un método de carga. De forma empírica podemos suponer que una mayor corriente carga mas rápido una batería, y esto es correcto, sin embargo, el sistema basado en algoritmos genéticos logra determinar esta afirmación por si solo. Los resultados expuestos concluyen que, como se hace mención en la sección anterior, una buena estructura de hardware facilita la implementación del experimento. El método de carga propuesto en comparación con el método de carga tradicional ofrece la posibilidad de explorar en un mayor campo de búsqueda las implicaciones y repercusiones que los parámetros de carga pueden afectar a una batería. Gracias a los resultados anteriores podemos concluir que existe un punto intermedio en donde el proceso de carga se vuelve mas eficiente, esto realizando una comparación entre tiempo de carga y la energía retenida por la batería. Un menor tiempo de carga pasando este punto intermedio converge en una menor retención de energía por parte de la batería, entonces sacrificamos tiempo de duración de una batería por un menor tiempo de carga. En el mejor de los casos

de la implementación del método de carga propuesto a una corriente de 1C se logro mejorar el tiempo de carga en un 85 % en comparación con el dispositivo tradicional de carga, y en un 93.25 % cuando se utilizaba una corriente de 2C. Una carga rápida tiene un impacto directo en la cantidad de energía que retiene una batería, aunque este no sea muy significativo, existe y se demostró en las gráficas expuestas en la sección anterior. Los resultados además de presentar la eficiencia del método de carga propuesto expresan diversos factores y comportamientos que no se tomaron en cuenta a un inicio del experimento y sin embargo son importantes, como lo es la retención de energía de una batería en comparación con el tiempo de carga, el análisis de la etapa de carga CV y las tendencias de evolución de los parámetros de carga en base a la función objetivo. Estas nuevas características pueden utilizarse como base de una función derivable y utilizar otra técnica de inteligencia artificial, por ejemplo una basada en gradiente.

9. Referencias

Referencias

[Atzori et al., 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of Things : A survey. (May).

[Ayoub and Karami, 2015] Ayoub, E. and Karami, N. (2015). Review on the charging techniques of a Li-Ion battery. *2015 Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pages 50–55.

[Bianchi et al., 2009] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*.

[Corp, 2015] Corp, N. T. P. A. (2015). TP4056 1A Standalone Linear Li-lon Battery Charger with Thermal Regulation in SOP-8.

[Coughlin, 2017] Coughlin, T. (2017). Market Trends are Driving Digital Storage Choices for Consumer Devices [The Art of Storage]. *IEEE Consumer Electronics Magazine*.

[Crompton and TR, 2000] Crompton and TR, editors (2000). *Battery Reference Book*. Newnes, third edition.

[Dps, 2019] Dps, D. P. S. (2019). DPS Series Constant Voltage / Constant Current Digital Power Supply Introductions. pages 17–32.

[Fogel, 2004] Fogel, D. B. (2004). Handbook of Evolutionary Computation. *Handbook of Evolutionary Computation*.

[Ghossein et al., 2015] Ghossein, N. E., Salameh, J. P., Karami, N., Hassan, M. E., and Najjar, M. B. (2015). Survey on Electrical Modeling Methods Applied on Different Battery Types. pages 39–44.

[Gibbard, 1994] Gibbard, H. F. (1994). Nickel Metal Hydride Battery Applications. page 1.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*.

[Goldberg and Deb, 1991] Goldberg, D. E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. page 27.

[Guzhan et al., 1998] Guzhan, O. ., Ëebi, H., and Erbatur, F. (1998). Evaluation of crossover techniques in genetic algorithm based optimum structural design. page 14.

[Haverkort, 2009] Haverkort, M. R. J. B. R. (2009). Which battery model to use ? (May).

[Heiss-Czedik, 2009] Heiss-Czedik, D. (2009). An Introduction to Genetic Algorithms. *Artificial Life*, 3(1):63–65.

[Hill, Jonh, 1999] Hill, Jonh, W. (1999). Química para el nuevo milenio. *Prentice hall*.

[Horiba, 2014] Horiba, B. T. (2014). Lithium-Ion Battery Systems. pages 1–12.

[Ikeya et al., 2002] Ikeya, T., Sawada, N., Murakami, J. I., Kobayashi, K., Hattori, M., Murotani, N., Ujiie, S., Kajiyama, K., Nasu, H., Narisoko, H., Tomaki, Y., Adachi, K., Mita, Y., and Ishihara, K. (2002). Multi-step constant-current charging method for an electric vehicle nickel/metal hydride battery with high-energy efficiency and long cycle life. *Journal of Power Sources*, 105(1):6–12.

[INEGI and ENDUTIH, 2018] INEGI and ENDUTIH (2018). COMUNICADO DE PRENSA NÚM . 105. volume 1, page 17.

[Jinghui Zhong et al., 2006] Jinghui Zhong, Xiaomin Hu, Jun Zhang, and Min Gu (2006). Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms. 1:1115–1121.

[Konak et al., 2006] Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91(9):992–1007.

[Leens, 2009] Leens, F. (2009). An introduction to I2C and SPI protocols. *IEEE Instrumentation & Measurement Magazine*, 12(1):8–13.

[Lin, 2009] Lin, C. (2009). An Adaptive Genetic Algorithm based on Population Diversity strategy.

[Lin et al., 2008] Lin, C. H., Chen, C. L., Lee, Y. H., Wang, S. J., Hsieh, C. Y., Huang, H. W., and Chen, K. H. (2008). Fast charging technique for Li-Ion battery charger. In *Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008*.

[Liu et al., 2016] Liu, W., Sun, X., Wu, H., He, Z., and Yang, G. (2016). A multistage current charging method for Li-ion battery bank considering balance of internal consumption and charging speed. *2016 IEEE 8th International Power Electronics and Motion Control Conference, IPEMC-ECCE Asia 2016*, (2):1401–1406.

[Mallick et al., 2016] Mallick, T. C., Ariful, M., Bhuyan, I., and Munna, M. S. (2016). Design & Implementation of an UAV (Drone) with Flight Data Record.

[Man et al., 1996] Man, K. F., Tang, K. S., and Kwong, S. (1996). Genetic algorithms: Concepts and applications. *IEEE Transactions on Industrial Electronics*, 43(5):519–534.

[MicroChip, 2007] MicroChip (2007). Mcp413x/415x/423x/425x.

[Mitchell, 2013] Mitchell, M. (2013). Genetic algorithms: An overview. *Complexity*, 1(1):31–39.

[Negnevitsky, 2005] Negnevitsky, M. (2005). *Artificial Intelligence*. Addisson-Wesley, Harlow, England, 1 edition.

[NELSON and BOLIN, 1995] NELSON, J. P. and BOLIN, W. D. (1995). Basics and Advances in Battery Systems (Vol 31, Pg 419, 1995). *Ieee Transactions on Industry Applications*, 31(4):725.

[Notten et al., 2005] Notten, P. H., Veld, J. H. H., and Van Beek, J. R. (2005). Boostcharging Li-ion batteries: A challenging new charging concept. *Journal of Power Sources*, 145(1):89–94.

[Pencheva et al., 2009] Pencheva, T., Atanassov, K., and Shannon, A. (2009). Modelling of a Stochastic Universal Sampling Selection Operator in Genetic Algorithms Using Generalized Nets. *Tenth Int. Workshop on Generalized Nets Sofia*, (December):1–7.

[Salameh and Kim, 2009] Salameh, Z. M. and Kim, B. G. (2009). Advanced lithium polymer batteries. *2009 IEEE Power and Energy Society General Meeting, PES '09*, pages 1–5.

[Tar and Fayed, 2016] Tar, B. and Fayed, A. (2016). An Overview of the Fundamentals of Battery Chargers. page 4.

Apéndice A

Apéndice A.1 Programa Arduino

```
#include <ModbusMaster.h>
#include <avr/interrupt.h>

int adcPort = 1;
int portOnOff = 9;
float voltage = 0;
float current = 0;
float power = 0;
float SetV = 0;
float SetA = 0;
bool on = 0;
uint8_t j, result;
int readComplete = 0;
char incomingByte;
unsigned long time;

// Instantiate ModbusMaster object
ModbusMaster node;

void readSerial() {

  result = node.readHoldingRegisters(0, 10); // slave: read a range
     ↪ of 16-bit registers starting at register 0 to 10

  if (result == node.ku8MBSuccess){ // only do something with data
     ↪ if read is successful
    voltage = ((float)node.getResponseBuffer(2) / 100 ); // get
       ↪ voltage from response buffer and convert to float
```

```
    current = ((float)node.getResponseBuffer(3) / 1000 ); // get
        ↪ current from response buffer and convert to float
    power = ((float)node.getResponseBuffer(4) / 100 ); // get power
        ↪ from response buffer and convert to float
    SetV = ((float)node.getResponseBuffer(0) / 100 ); // get SetV
        ↪ from response buffer and convert to float
    SetA = ((float)node.getResponseBuffer(1) / 1000 ); // get SetA
        ↪ from response buffer and convert to float
    on = ((bool)node.getResponseBuffer(9) ); // Status on or off

    Serial.print(voltage);
    Serial.print(",");
    Serial.print(current);
    Serial.print(",");
    Serial.print(power);
    Serial.print(",");
    Serial.print(SetV);
    Serial.print(",");
    Serial.print(SetA);
    Serial.print(",");
    Serial.print(on);
    Serial.println();
  }
}

void writeSerial(int port, int value) {
  node.writeSingleRegister(port,value);
}

void setPowerSupply(float voltage, float current){
  node.writeSingleRegister(0,voltage*100);
  node.writeSingleRegister(1,current*1000);
```

```
}

void setVoltage(float voltage){
  node.writeSingleRegister(0,voltage*100);
}

void setCurrent(float current){
  node.writeSingleRegister(1,current*1000);
}

void setup() {
  DDRB|=1<<PB7;
  DDRE|=1<<PB4;
  ADMUX = (1<<REFS0) | (1<<MUX0); // Vref, ADC1
  ADCSRA = (1<<ADEN) | (7<<ADPS0); //125KHz, Enable
  ADCSRB = 0;
  ADCSRA |= (1<<ADIE)|(1<<ADATE);
  ADCSRA |= (1<<ADSC);
  // use Serial (port 0); initialize Modbus communication baud rate
  Serial.begin(19200);
  Serial2.begin(19200);
  // communicate with Modbus slave ID 1 over Serial (port 0)
  node.begin(1, Serial2);
}

void loop() {
  int thisTime = 0;
  int x = 0;

  time = micros();
  writeSerial(portOnOff,1);
  thisTime = (micros()-time)/1000;
```

```
  Serial.println("Tiempo␣para␣inicializar:");
  Serial.println(thisTime);

  while(1) {
    for(x=0;x<30;x++) {
      time = micros();
      PORTB^=1<<PB7;
      PORTE |= 1<<PE4;
      setVoltage(1);
      PORTB^=1<<PB7;
      thisTime = (micros()-time)/1000;
      Serial.println(thisTime);
      delay(x*50);
      time = micros();
      PORTB^=1<<PB7;
      PORTE |= 1<<PE4;
      setVoltage(3);
      PORTB^=1<<PB7;
      thisTime = (micros()-time)/1000;
      Serial.println(thisTime);
      delay(x*50);
      time = micros();
    }
  }
}

ISR(ADC_vect) {
  static int result;
  result = ADCL;
  result |= (ADCH<<8);
  if(result>=410) {
    PORTE &= !(1<<PE4);
```

```
  }
}
```

Apéndice A.2 Fuente programable

```
import minimalmodbus

class PowerSupply:

voltageRegisterw = 0
currentRegisterw = 1
voltageRegisterr = 2
currentRegisterr = 3
turnPower = 9

def __init__(self,port,baudrate,slave):
        self.instrument = minimalmodbus.Instrument(port,slave,'rtu')
            ↪ # port name, slave address (in decimal)
        self.instrument.serial.baudrate = baudrate
        self.instrument.serial.timeout = 2 # default seconds
        self.instrument.write_register(self.voltageRegisterw, 10*5, 1)
            ↪  # Registernumber, value, number of decimals for
            ↪ storage

def setVoltage(self,voltage):
        self.instrument.write_register(self.voltageRegisterw, voltage
            ↪ *10, 1) # Registernumber, value, number of decimals
            ↪ for storage

def getVoltage(self):
        return self.instrument.read_register(self.voltageRegisterr)
            ↪ /100
```

```
def getCurrent(self):
        return self.instrument.read_register(self.currentRegisterr)
            ↪ /1000

def setCurrent(self,current):
        self.instrument.write_register(self.currentRegisterw, current
            ↪ /10, 1) # Registernumber, value, number of decimals
            ↪ for storage

def turnOnPower(self):
        self.instrument.write_register(self.turnPower, 1) #
            ↪ Registernumber, value, number of decimals for storage

def turnOffPower(self):
        self.instrument.write_register(self.turnPower, 0) #
            ↪ Registernumber, value, number of decimals for storage
```

Apéndice A.3 Algoritmo Genético

```
from random import randint
import PowerSupply
import time
import serial
import threading
import math

## Mask's for CC 2 steps [Bits to shift,mask,divisior] (2C)
get_topCurrent21 = [31,4395899027456,2]
get_lowCurrent21 = [20,2146435072,2]
get_topCurrent2 = [31,4395899027456,1]
get_lowCurrent2 = [20,2146435072,1]
```

```
get_topTime2 = [10,1047552,100]
get_lowTime2 = [0,1023,100]
mode21 = [get_topCurrent21,get_lowCurrent21,get_topTime2,get_lowTime2
    ↪ ]
mode2 = [get_topCurrent2,get_lowCurrent2,get_topTime2,get_lowTime2]

## Mask's for CC 3 steps [Bits to shift,mask](2C)
get_topCurrent31 = [52,9218868437227405312,2]
get_midCurrent31 = [41,4501400604114944,2]
get_lowCurrent31 = [30,2197949513728,2]
get_topCurrent3 = [52,9218868437227405312,1]
get_midCurrent3 = [41,4501400604114944,1]
get_lowCurrent3 = [30,2197949513728,1]
get_topTime3 = [20,1072693248,100]
get_midTime3 = [10,1047552,100]
get_lowTime3 = [0,1023,100]
mode31 = [get_topCurrent31,get_midCurrent31,get_lowCurrent31,
    ↪ get_topTime3,get_midTime3,get_lowTime3]
mode3 = [get_topCurrent3,get_midCurrent3,get_lowCurrent3,get_topTime3
    ↪ ,get_midTime3,get_lowTime3]

def generateFirstPopulation(sizePopulation):
        r = 0
        population = []

        for x in range(sizePopulation):
                if chargeMode == 0:
                        population.append(randint(1, 4398046511103))
                elif chargeMode == 1:
                        population.append(randint(1,
                            ↪ 9223372036854775807))
```

```
        return population


def getValuesfromIndividuals(chromosome):
        data = []
        data2 = []
        divisor = 2


        if chargeMode == 0:
                if cRate == 1:
                        for x in range(len(mode21)):
                                data.append(getValuesFromGenes(
                                    ↪ chromosome,mode21[x])/mode21[x][
                                    ↪ divisor])
                                if x < 2:
                                        data[x] = math.ceil(data[x])
                elif cRate == 2:
                        for x in range(len(mode2)):
                                data.append(getValuesFromGenes(
                                    ↪ chromosome,mode2[x])/mode2[x][
                                    ↪ divisor])
        if chargeMode == 1:
                if cRate == 1:
                        for x in range(len(mode31)):
                                data.append(getValuesFromGenes(
                                    ↪ chromosome,mode31[x])/mode31[x][
                                    ↪ divisor])
                                if x < 3:
                                        data[x] = math.ceil(data[x])
                if cRate == 2:
                        for x in range(len(mode3)):
                                data.append(getValuesFromGenes(
                                    ↪ chromosome,mode3[x])/mode3[x][
```

```
                                    ↪ divisor])

        return data

def getIndividualsfromValues(values):
        bitstoshift = 0
        chromosome = 0

        if chargeMode == 0:
                for x in range(len(mode2)):
                        chromosome |= values[x]<<mode2[x][bitstoshift]
        if chargeMode == 1:
                for x in range(len(mode3)):
                        chromosome |= values[x]<<mode3[x][bitstoshift]

        return chromosome



def getValuesFromGenes(chromosome,parameter):
        bitstoshift = 0
        mask = 1

        return ((chromosome & parameter[mask]) >> parameter[
            ↪ bitstoshift])

def getPopulationValues(population):
        populationValues = []

        for x in range(len(population)):
                populationValues.append(getValuesfromIndividuals(
                    ↪ population[x]))
```

```
        return populationValues


def fitnessFunction(individuals):
        poblation = []
        contextFitness = []
        individual = 0
        time = 1
        x = 0
        addition = 0
        global bestbefore
        #Global Poblation
        for i in range(populationSize):
                #Evaluate with subtraction
                poblation.append([individuals[i][individual],
                    ↪ globalFitnessValue - individuals[i][time]])
                #Record the entire poblation
                globalPoblation.append(poblation[i])
                #Find the best of the entire poblation
                if poblation[i][time] >= poblation[x][time]:
                        x = i
        print("Global Poblation: ",globalPoblation)

        #Best Global
        bestGlobal.append(poblation[x])
        f.write("El mejor de la poblacion: "+str(poblation[x]))

        #Context Fitness - Addition
        for j in range(populationSize):
                addition += individuals[j][time]

        del poblation[:]
```

```
        #Context Fitness - Divition
        for k in range(populationSize):
                poblation.append([individuals[k][individual],addition/
                    ↪ individuals[k][time]])

        return sorted(poblation, key=lambda poblation: poblation[1],
            ↪ reverse=True) # sort by age

def createWheel(individuals):
        init = 0
        rank = 1
        individual = 0
        wheel = []

        for x in range(len(individuals)):
                wheel.append([init,init+individuals[x][rank],
                    ↪ individuals[x][individual]])
                init += individuals[x][rank]

        return wheel

def wheelSearch(individuals,fixedPoints):
        flag = 0
        fpp = 0 #Fixed Point Pointer
        selection = []
        wheel = createWheel(individuals)
        print(individuals)

        if elitism == 1:
                low, high,answer = wheel[0]
                selection.append(answer)
                del(individuals[0])
```

```
                wheel = createWheel(individuals)
                fpp += 1
                fixedPoints[fpp] -= high

        while fpp != len(fixedPoints):
                for i in range(len(wheel)):
                        low, high,answer = wheel[i]
                        if low<=fixedPoints[fpp]<=high:
                                selection.append(answer)
                                del(individuals[i])
                                wheel = createWheel(individuals)
                                fpp += 1
                                flag = 1
                                break
                if flag != 1:
                        fixedPoints[fpp] -= high
                else:
                        flag = 0

        print("Selection:␣",selection)
        f.write("Cromosomas␣seleccionados:␣"+str(selection)+"\n")
        f.write("Valor␣de␣Cromosomas␣seleccionados:␣")
        for v in range (len(selection)):
                f.write(str(getValuesfromIndividuals(selection[v])))
        f.write("\n")
        return selection

def SUSSelection(individuals):
        fixedPoints = []

        #Get fixed points
        for x in range(int(len(individuals)/2)):
```

```
                fixedPoints.append(randint(1,200))

        return wheelSearch(individuals,fixedPoints)

def reproduction(chromosomes):
        #For x point crossover
        definedTimePoints = [0,3,7,10]
        definedCurrentPoints = [0,3,9,11]
        definedPoints = [definedCurrentPoints,definedTimePoints]
        #Mode: 0 - Top Current; 1 - Low Current; 2 - Top Time; 3 -
            ↪ Low Time
        mode = 2
        #This value depends on charging mode technique
        if chargeMode == 0:
                params = 2
        else:
                params = 3
                order=[2,1,0,4,3,2]
        #Offspring size calc
        oSize = int(populationSize/2)
        offspring = []
        offspringChromosomes = []
        for i in range(populationSize):
                offspring.append([])
        regularcroppedSubstrings = []

        #Iterative process
        for q in range(oSize):
                for y in range(2):
                        for x in range(params):#Crossover Section
                                regularcroppedSubstrings =
                                    ↪ regularCrossoverSort(
```

```
                                    ↪ crossoverPoints(chromosomes,
                                    ↪ definedPoints[y],(x+params*y)))
                                for z in range(oSize):#Mutate Section
                                        offspring[z+oSize*q].append(
                                            ↪ substringToValue(
                                            ↪ regularcroppedSubstrings[z
                                            ↪ ],definedPoints[y]))
                                        mutateIndiviual(offspring[z+oSize
                                            ↪ *q],y)
                    chromosomes[q]=getIndividualsfromValues(
                        ↪ reorderChromosome(chromosomes[q]))

        for a in range(len(offspring)):
                    offspringChromosomes.append(getIndividualsfromValues(
                        ↪ offspring[a]))

        print(offspringChromosomes)
        f.write("Nueva␣generacion:␣" + str(offspringChromosomes)+"\n"
            ↪ )

        return offspringChromosomes


def reorderChromosome(chromosome):
        chrom = []

        if chargeMode == 0:
                    order = [1,0,3,2]
                    for z in range(len(mode2)):
                            chrom.append(getValuesFromGenes(chromosome,
                                ↪ mode2[z]))
                    return [ chrom[i] for i in order]
```

```
        elif chargeMode == 1:
                order=[2,1,0,4,3,2]
                for z in range(len(mode3)):
                        chrom.append(getValuesFromGenes(chromosome,
                            ↪ mode3[z]))
                return [ chrom[i] for i in order]

def crossoverPoints(chromosomes,points,mode):
        croppedSubstring1 = []
        croppedSubstring2 = []
        croppedSubstring = [croppedSubstring1,croppedSubstring2]
        substrings = []

        for x in range(len(chromosomes)):
                if chargeMode == 0:
                        substrings.append(getValuesFromGenes(
                            ↪ chromosomes[x],mode2[mode]))
                elif chargeMode == 1:
                        substrings.append(getValuesFromGenes(
                            ↪ chromosomes[x],mode3[mode]))

        for y in range(len(points)-1):
                substring1,substring2 = getAndCutSubstring(
                    ↪ croppedSubstring,substrings,[points[y],points[y
                    ↪ +1]])

        return croppedSubstring

def regularCrossoverSort(croppedSubstrings):
        offspring1 = []
        offspring2 = []
```

```
        for x in range(len(croppedSubstrings[0])):
                offspring1.append(croppedSubstrings[x%2][x])
                offspring2.append(croppedSubstrings[~(x%2)+2][x])

        return [offspring1,offspring2]

def substringToValue(substring,points):
        value = 0
        size = len(points)-1

        for x in range(size):
                value |= substring[size-x-1]<<points[size-x-1]

        return value

def getAndCutSubstring(croppedSubstrings,substrings,points): #
    ↪ Ascending points order

        for x in range(len(croppedSubstrings)):
                croppedSubstrings[x].append(substrings[x]&
                    ↪ arithmeticShiftbyOne(points[1]-points[0]))
                substrings[x] = substrings[x] >> points[1]-points[0]

        return substrings


def arithmeticShiftbyOne(shifts):
        number = 1

        for x in range(shifts-1):
                number = number << 1
                number = number + 1
```

```
        return number

def mutateIndiviual(individual,type): #Type possible values: 0 -
    ↪ Time ; 1 - Current
        bits = 10 - type

        for x in range(len(individual)):
                if(randint(0, 100)<(mutationRate*100)):
                        individual[x] ^= 1<<randint(0,bits)

def recolectDataCC():
        newstr = ""
        aux = ""
        flag = 0
        global finishThread
        global chragedischarge
        global voltage

        f.write("Voltage:␣")
        if chragedischarge == 0:
                s.write(b'c')
        elif chragedischarge == 1:
                s.write(b'd')
                print("Holo")

        while(newstr != 'f'):
                data = s.readline()
                newstr = str(data).replace("b", "").replace("'","").
                    ↪ replace("r","").replace("n","").replace('\\\\','
                    ↪ ')
                print(newstr)
```

```
                if "." in newstr:
                        f.write(newstr+"\t")
                aux = newstr[0:4]
                print(aux)
                if flag == 0 and chragedischarge == 0:
                        gChargeAmount.append(aux)
                        flag = 1
                try:
                        voltage = float(aux)
                        print("true")
                except:
                        print("False")

        f.write("\n")
        print("Recolect Done.")
        finishThread = 1

def recolectDataCV():
        newstr = ""
        aux = ""
        global finishThread
        global voltage

        s.write(b'c')
        print("\nCV Mode ON\n")

        while(finishThread == 0):
                data = s.readline()
                newstr = str(data).replace("b", "").replace("'","").
                    ↪ replace("r","").replace("n","").replace('\\\\','
                    ↪ ')
                print(newstr)
```

```
                if "." in newstr:
                        f.write(newstr+"\t")
                aux = newstr[0:4]
                print(aux)
                try:
                        voltage = float(aux)
                        print("true")
                except:
                        print("False")
        s.write('s')
        f.write("\n")
        print("Recolect␣Done.")
        finishThread = 1

def testIndividuals(chromosomes):
        individualsGrade = []
        times = []
        current = []
        result = []
        global finishThread
        global chragedischarge

        for x in range(len(chromosomes)):
                #Charging mode
                f.write("Carga\n")
                individualsGrade.append(chromosomes[x])
                print(individualsGrade)
                if chargeMode == 0:
                        current = [getValuesfromIndividuals(chromosomes
                            ↪ [x])[0],getValuesfromIndividuals(
                            ↪ chromosomes[x])[1]]
```

```
            times = [getValuesfromIndividuals(chromosomes[x
                ↪ ])[2],getValuesfromIndividuals(
                ↪ chromosomes[x])[3]]

            print("Corriente:␣",current,"␣Tiempos:␣",times)
            f.write("Corriente:␣"+str(current)+"␣Tiempos:␣"
                ↪ +str(times))

            tiempo = ccMode(current,times)
            print("Tiempo␣CC:␣", ("%.2f" % ((tiempo)/60)))
            f.write("\nTiempo␣CC:␣"+ ("%.2f" % ((tiempo)
                ↪ /60)))

            #Generate info
            gChromosomes.append(individualsGrade)
            gCurrentParams.append(current)
            gTimeParams.append(times)
            gChargeTimes.append(("%.2f" % ((tiempo)/60)))

        elif chargeMode == 1:
            current = [getValuesfromIndividuals(chromosomes
                ↪ [x])[0],getValuesfromIndividuals(
                ↪ chromosomes[x])[1],
                ↪ getValuesfromIndividuals(chromosomes[x])
                ↪ [2]]
            times = [getValuesfromIndividuals(chromosomes[x
                ↪ ])[3],getValuesfromIndividuals(
                ↪ chromosomes[x])[4],
                ↪ getValuesfromIndividuals(chromosomes[x])
                ↪ [5]]
            print("Corriente:␣",current,"␣Tiempos:␣",times)
```

```
            f.write("Corriente: "+str(current)+" Tiempos: "
                ↪ +str(times))

            tiempo = ccMode(current,times)
            print("Tiempo CC: ", ("%.2f" % ((tiempo)/60)))
            f.write("\nTiempo CC: "+ ("%.2f" % ((tiempo)
                ↪ /60)))

            #Generate info
            gChromosomes.append(individualsGrade)
            gCurrentParams.append(current)
            gTimeParams.append(times)
            gChargeTimes.append(("%.2f" % ((tiempo)/60)))

    print("Carga Finalizada.")
    finishThread = 0
    individualsGrade.append(float(("%.2f" % ((tiempo)/60))
        ↪ ))
    print(individualsGrade)
    result.append(individualsGrade)
    individualsGrade = []
    energy.turnOffPower()
    time.sleep(sleepTime)
    print(result)

    #Discharging mode
    f.write("Descarga\n")
    tiempo = time.time()
    chragedischarge = 1
    t1 = threading.Thread(target=recolectDataCC)
    t1.start()
    while finishThread == 0:
```

```
                pass
            finishThread = 0
            print("Tiempo de descarga: "+("%.2f" % ((time.time() -
                ↪ tiempo)/60)))
            f.write("Tiempo de descarga: "+("%.2f" % ((time.time()
                ↪ - tiempo)/60)))

            #Generate info
            gDischargeTimes.append(("%.2f" % ((time.time() -
                ↪ tiempo)/60)))
            time.sleep(sleepTime)

            #Record info
            g.write(str(gChargeAmount[x+(generation*4)])+"\n")
            g.write(str(gChromosomes[x+(generation*4)])+"\t")
            g.write(str(gCurrentParams[x+(generation*4)])+"\t")
            g.write(str(gTimeParams[x+(generation*4)])+"\t")
            g.write(str(gChargeTimes[x+(generation*4)])+"\t")
            g.write(str(gDischargeTimes[x+(generation*4)])+"\t")

        print (result)
        suma = 0
        for m in range(len(result)):
            suma += result[m][1]
        print("Suma de los tiempos: ",suma)
        f.write("\n Fitness de toda la poblacion: "+str(suma))
        return result

def ccMode(current, times):
        global finishThread
        global chragedischarge
        global voltage
```

```
        energy.turnOnPower()
        chragedischarge = 0
        t1 = threading.Thread(target=recolectDataCC)
        t1.start()
        tiempo = time.time()

        while finishThread == 0:
                print("Voltaje en flotante: ",voltage," Tipo: ",type(
                    ↪ voltage),"\n")
                energy.setCurrent(current[0])
                time.sleep(times[0])
                energy.setCurrent(current[1])
                time.sleep(times[1])
                if(chargeMode == 1):
                        energy.setCurrent(current[2])
                        time.sleep(times[2])

        print("CC Mode Finalizada.")
        finishThread = 0
        energy.turnOffPower()

        return time.time() - tiempo

def cvMode():
        global finishThread
        global chragedischarge

        energy.turnOnPower()
        chragedischarge = 0
        t1 = threading.Thread(target=recolectDataCC)
        t1.start()
```

```
        tiempo = time.time()
        energy.setVoltage(4.4)
        finishThread = 0

        while finishThread == 0:
                energy.setVoltage(4.2 + (4.2-voltage))
                print("Voltaje␣en␣flotante:␣",voltage,"␣Tipo:␣",type(
                    ↪ voltage),"\n")
                print("Corriente:␣",energy.getCurrent())

        print("CV␣Mode␣Finalizada.")
        finishThread = 0
        energy.turnOffPower()
        energy.setVoltage(10)

        return time.time() - tiempo

def takeSecond(elem):
        return elem[1]

def getBestIndividual(array):
        arrayaux = array

        arrayaux.sort(reverse=True,key=takeSecond)

        return arrayaux[0]

#Main program

#Set up
energy = PowerSupply.PowerSupply('COM4',9600,1)
s = serial.Serial('COM8',baudrate=19200,timeout=1)
```

```
f= open("data.txt","w+")
g= open("dataresume.txt","w+")
gChromosomes = [1]
gCurrentParams = [2]
gTimeParams = [3]
gChargeTimes = [4]
gDischargeTimes = [5]
gChargeAmount = [6]
finishThread = 0
chragedischarge = 0
time.sleep(2)
response = 'n'
sleepTime = 300
voltage = 0
flag = 0
bestbefore = []
num = 0

#Stadistics
globalPoblation = [] #This list contains all the poblations tested
bestGlobal = []#This list contains the best global individuals

#General Configuration
currentBits = 11
timeBits = 10
globalFitnessValue = 500
populationSize = 4
chargeMode = 1
mutationRate = 0.1
elitism = 1
population = []
epochs = 100
```

```
generation = 0
cRate = 1

while response == 'n':
        #Generate first poblation
        population = generateFirstPopulation(populationSize)
        f.write(str(population)+"\n")
        print(population)

        for y in range(len(population)):
                print(getValuesfromIndividuals(population[y]))

        response = input("Iniciar␣con␣esos␣Cromosomas?␣y/n")

population = [4345346029985, 2158193182622, 2141023791014,
    ↪ 2158201563046]
f.write(str(population)+"\n")

tiempo = time.time()
chragedischarge = 1
t1 = threading.Thread(target=recolectDataCC)
t1.start()
while finishThread == 0:
        pass
finishThread = 0
time.sleep(sleepTime)

while generation <= epochs:

        population = reproduction(SUSSelection(fitnessFunction(
            ↪ testIndividuals(population))))
        print(population)
```

```
        f.write("----------------------Generacion␣"+str(generation)+"␣
            ↪ ---------------------")
        f.write("\nNueva␣Poblacion:␣" + str(population))
        print("Generacion␣",generation,"\n")
        num =num + 1
        generation = num

f.write("\nDone")
print("Done")
f.close()
```