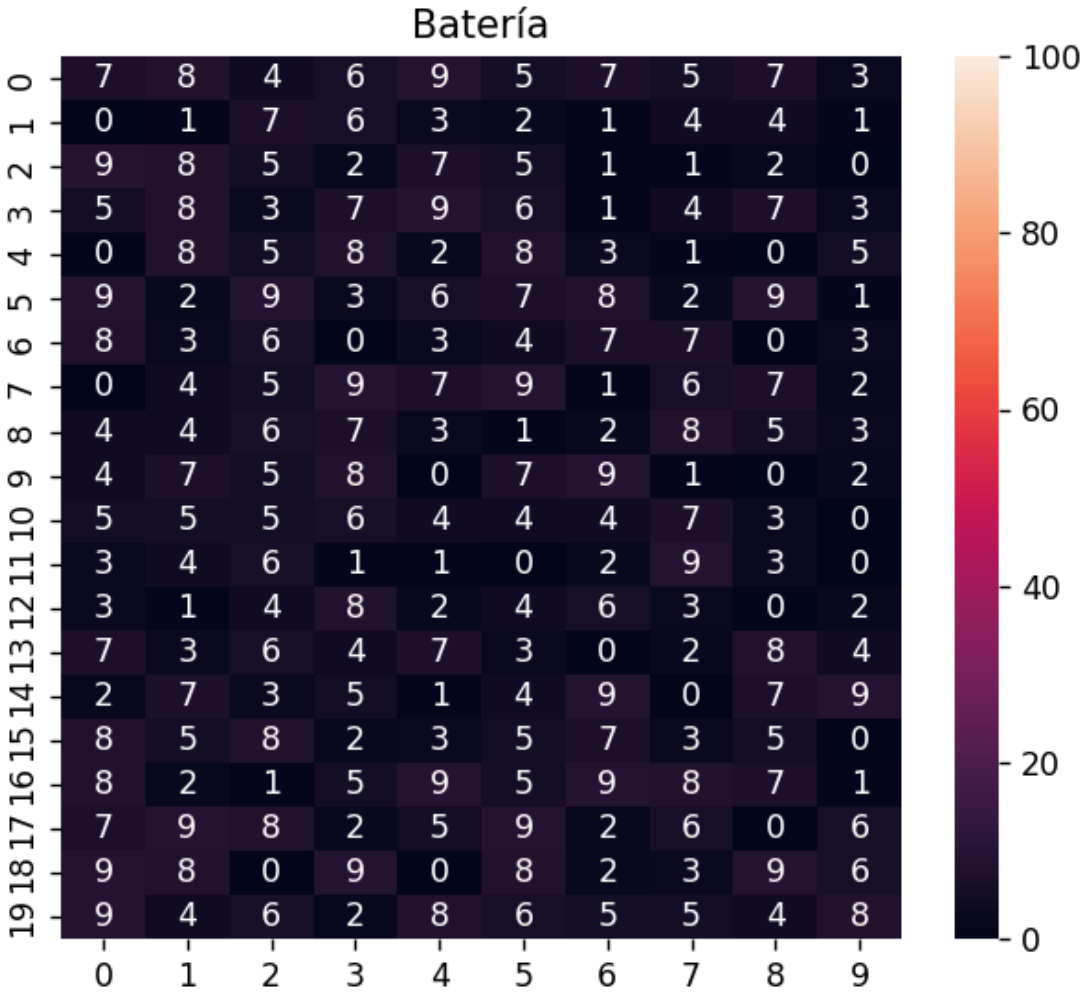
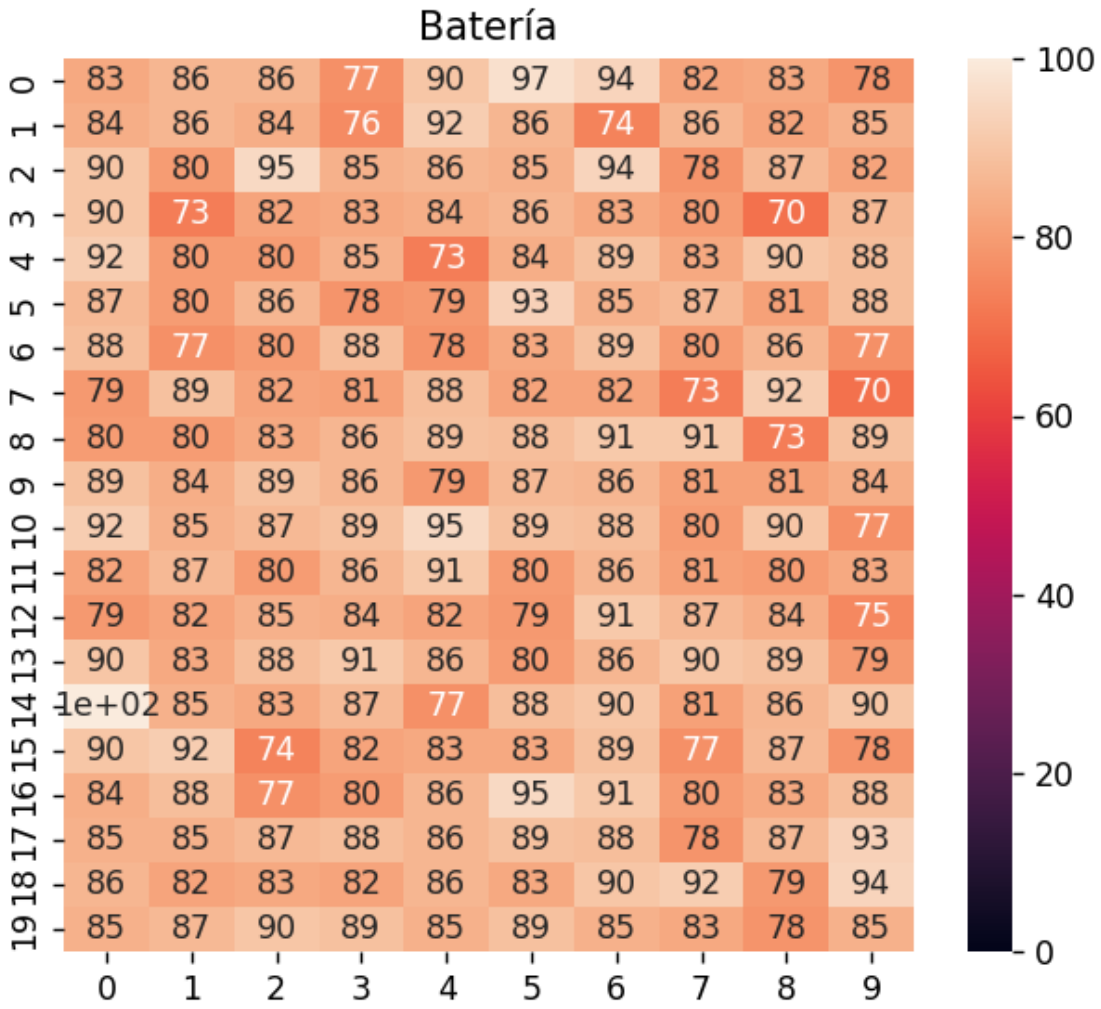
Descripción del problema

· Existe una matriz de tamaño (i, j) que representa los nodos colocados en el campo. Cada celda de la matriz contiene un entero [0, 100] que representa cuánta batería ha gastado e, inicialmente, consideramos un desgaste inicial aleatorio en el rango [0, 10] (**por representar que no todas las baterías están totalmente cargadas al inicio, este aspecto se puede cambiar en la simulación**).

Un ejemplo de estado inicial para una instancia de tamaño (20, 10) sería la siguiente;



El objetivo es lograr que, al finalizar la simulación, **todas las celdas estén lo más cerca posible de haber gastado su energía total** (es decir, llegar a 100). El criterio de parada de la simulación se da cuando existe una batería queda totalmente gastada (la celda llega a un valor de 100) (**uso este criterio ya que representa el momento en el que el encargado de gestión tendrá que renovar al menos esa batería que se ha gastado**).



Ejemplo de estado final. Se puede observar que un único nodo alcanzó agotar su energía (la celda [14,0] vale 100) mientras que el resto alcanzó otros niveles más bajos (**se optó por la representación sobre un mapa de calor para leer mas facilmente el output**).

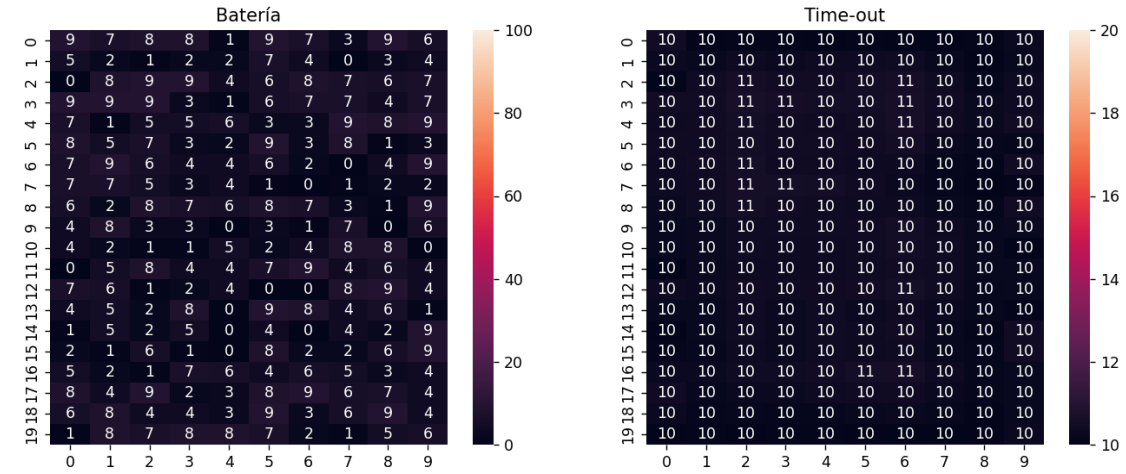
Condiciones de la simulación

Decimos que un nodo **gasta una cantidad aleatoria de energía** acotada por el rango [BAT_DRAIN_min, BAT_DRAIN_max] **cada vez que ejecuta una medición**, y se le debe de **asignar un tiempo de espera** en el rango [T_LIMIT_min, T_LIMIT_max] (**nomenclatura siguiendo la usada en el código**).

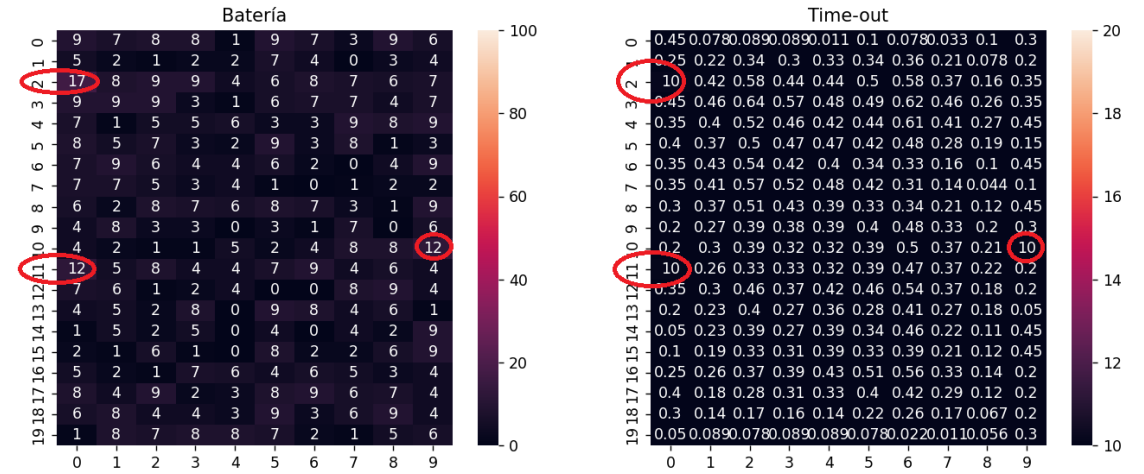
Para representar qué tiempo de espera se le ha asignado a cada nodo, existe una matriz (i,j) (_timeOut) con valores en el rango [T_LIMIT_min, T_LIMIT_max].

Al inicio de la simulación, el algoritmo debe asignar un tiempo de espera inicial (siempre dentro del rango definido) y simulará el paso del tiempo (**dicho valor del tiempo queda representado en _T**). Cada vez que un nodo cumple con su tiempo de espera, se le añade el

gasto de energía correspondiente (**un número aleatorio dentro del rango BAT_DRAIN**) y se vuelve a ejecutar el algoritmo para que le asigne un nuevo tiempo de espera (**nuevamente, dentro del rango T_LIMIT**).



Ejemplo de estado inicial, donde se considera que se pueden asignar tiempos de espera en el rango [10, 20] (**segundos**). A continuación, la simulación hará que transcurra el tiempo suficiente como para que **al menos un nodo simule que toma una medición** (**se le incrementa la cantidad de energía gastada en el proceso en su celda, en este caso para facilitar la visualización tomo que se gastará una cantidad aleatoria entre [10, 20]%**).



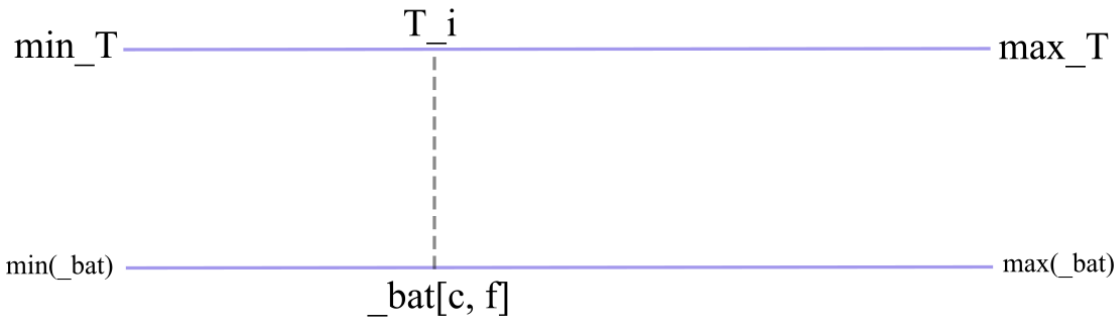
Tras pasar 10 segundos como hubieron 3 nodos con la misma espera, se activaron a la vez. Simulan el tomar una medida y se les incrementa la cantidad de energía gastada. Tras esto, como deben de ponerse nuevamente en pausa, se utiliza un algoritmo que les asigna la pausa entre [10, 20] segundos.

Cálculo de la asignación de tiempo

Esta forma de cálculo es la que se implementa en **'second.py'**, la cual es la que ofrece los mejores resultados. 'first.py' solo existe como una experiencia práctica secundaria relativa a mi método de clasificación.

Poseemos un rango mínimo y máximo de tiempos de espera **[min_T, max_T]** y la energía agotada por cada nodo (**_bat**) de lo cual se puede deducir un rango de valores de batería actual **[min(_bat), max(_bat)].** El objetivo actual es asignar un tiempo de espera idóneo a un nodo del conjunto (**de tal forma que se ajuste al nivel de batería del resto del conjunto**) (lo llamaremos **T_i**) y tomaremos como referente su energía gastada (**_bat[c, f]**).

Sabiendo que T_i es un valor dentro del rango de tiempos de espera y que la energía del nodo actual pertenece al rango de valores de batería, se puede observar una relación directa entre el tiempo de espera a asignar con la cantidad de batería que ha gastado el nodo actual.



Por ejemplo, si el nodo actual es el que ha gastado la menor cantidad de batería del conjunto, se le asignará la menor espera posible dentro del rango de posibles tiempos a asignar. De forma opuesta, si el nodo actual resulta ser aquel que ha gastado más batería del conjunto, se le dará el mayor tiempo de espera posible.

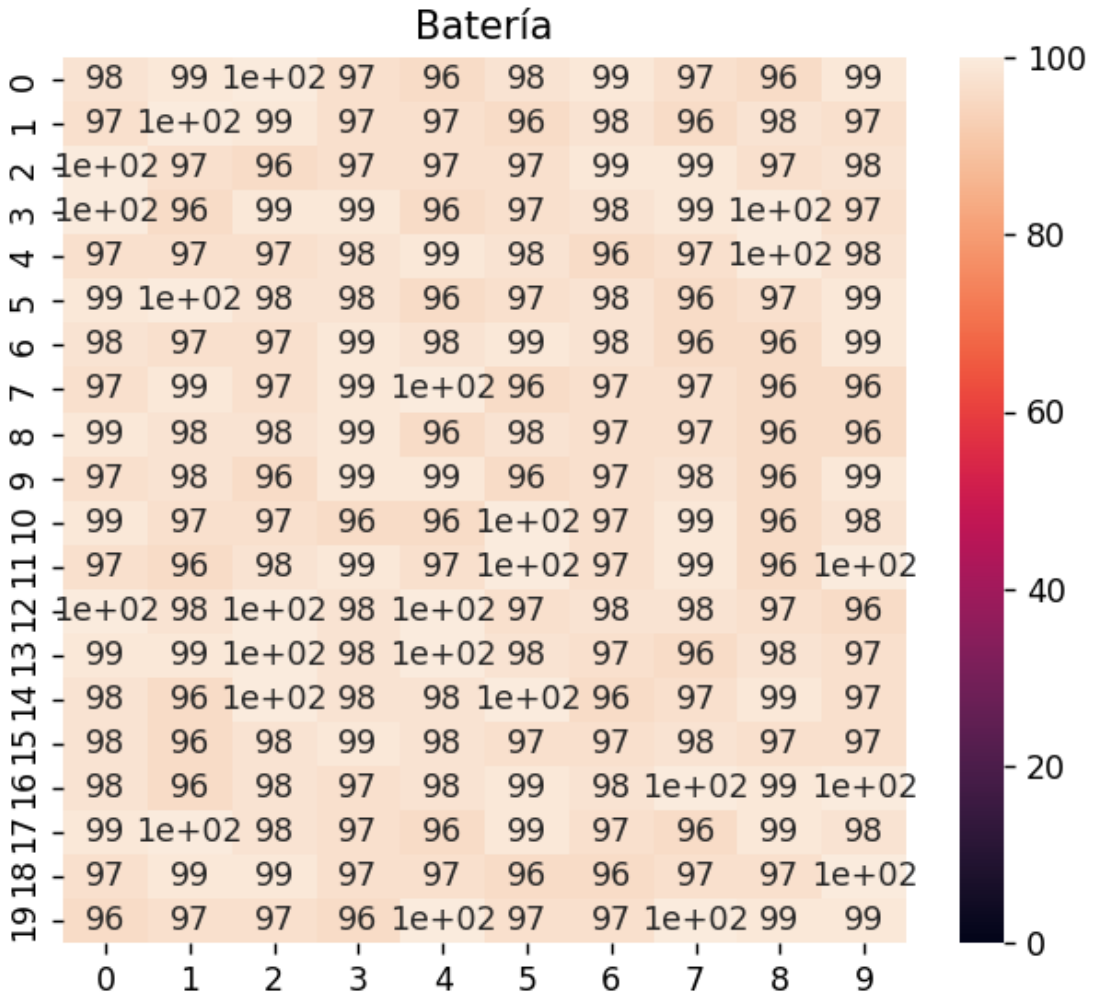
Este calculo se puede realizar con dos reglas de 3 (**donde 'c' se corresponde a '_bat' y ci a _bat[c, f]**). En la primera se haya donde se sitúa c_i en la escala de baterías (x%) y en la segunda se deduce el valor de T_i.

$$\left\{\begin{matrix} c_{max} & \rightarrow 100\,\% \\ c_i & \rightarrow x\,\% \end{matrix}\right\} \quad \left\{\begin{matrix} T_{max} - T_{min} & \rightarrow 100\,\% \\ T_i - T_{min} & \rightarrow x\,\% \end{matrix}\right\}$$

Motivo de la solución

El motivo principal es que **esta solución permite definir qué tiempos de espera se pueden asignar** de tal forma que no tenga repercusiones negativas en el resto del proyecto.

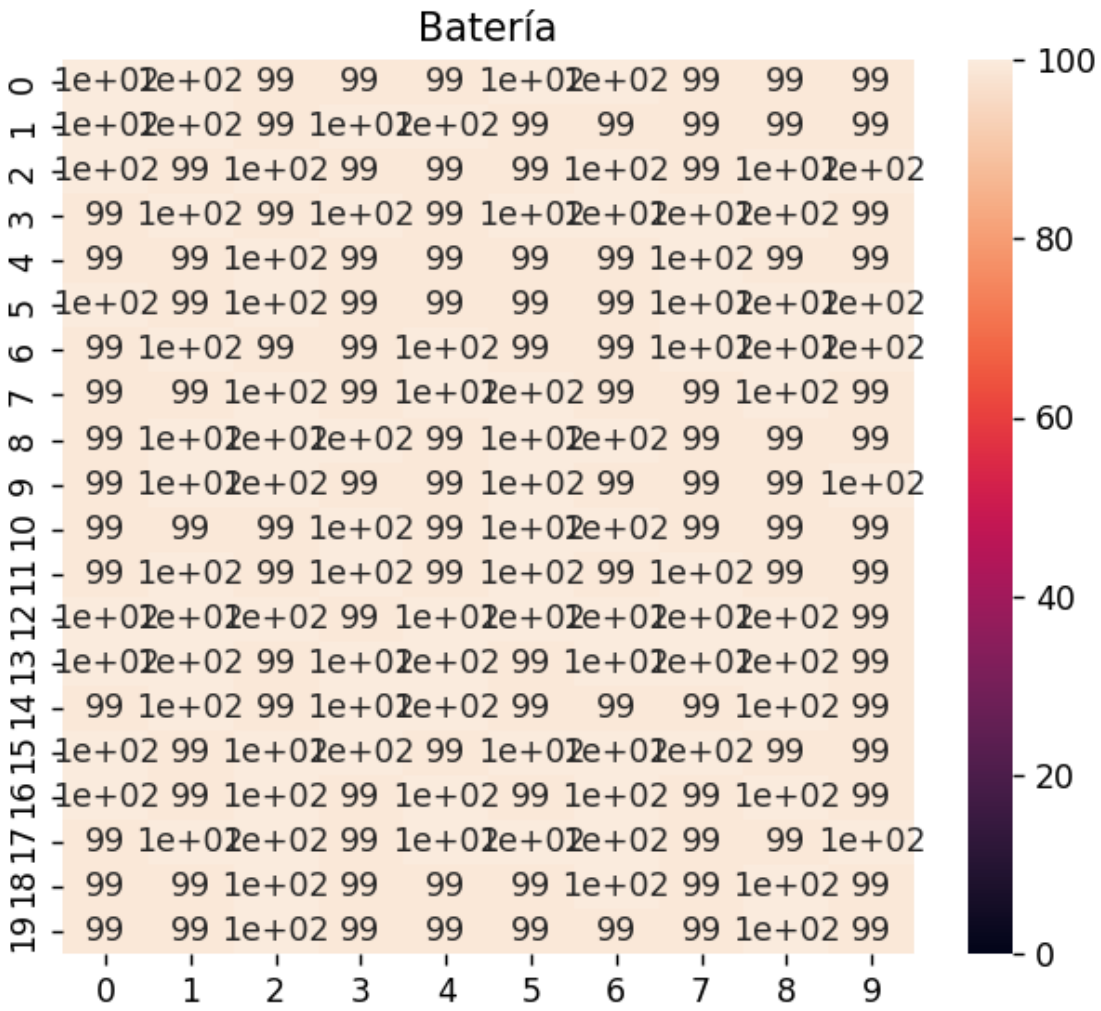
La disposición de los nodos no es relevante, y aun definiendo un rango pequeño de tiempos de espera asignables, los resultados siguen siendo aceptables;



Ejemplo de ejecucion, permitiendo tiempos de espera entre 10 y 13 minutos.

Esta solución solo utiliza un traspaso de escalas y con una programación adecuada (almacenando mínimos y máximos) se puede calcular qué tiempo asignar en O(1).

Finalmente resaltar que los ejemplos expuestos están considerando que las baterías pueden estar hasta un 10% de uso. Si consideramos un uso inicial de hasta un 2%, los resultados son mucho más positivos.



Relativo al código

En el primer bloque están declarados todos los parámetros y condiciones de la simulación, se pueden hacer pruebas con las condiciones que se consideren más realistas/extremistas.

En el código se presenta también una simulación del estado de las baterías sin hacer uso del método explicado (debería ser similar a la realidad si el gasto de energía queda bien definido).

Además para ver la evolución a lo largo del tiempo se puede descomentar la línea 149.