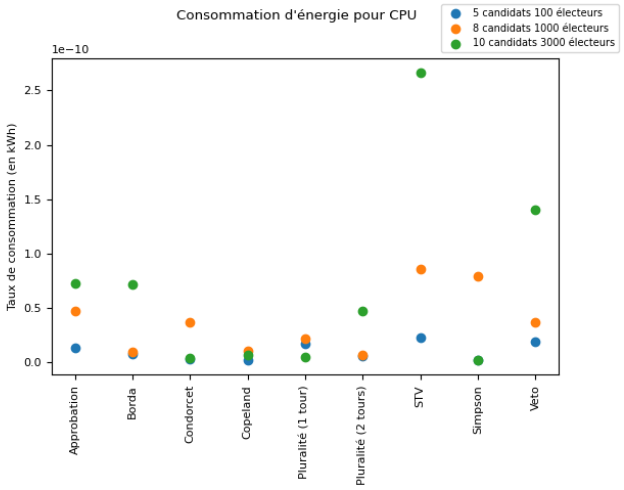
Rapport carbone

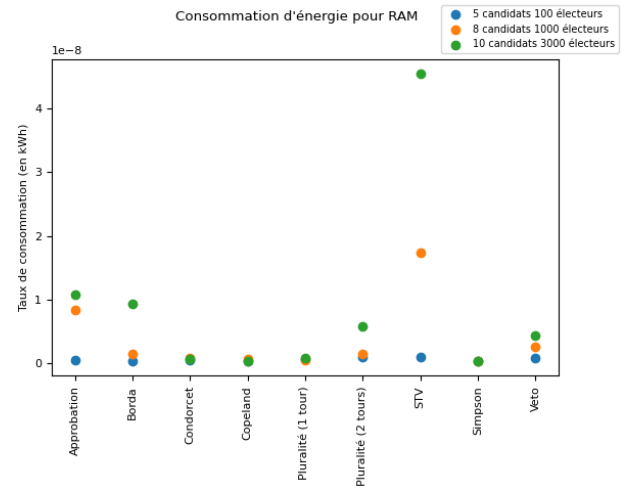
Pour effecteur des mesures l'outil `CodeCarbon` a été utilisé.

Dans notre code, les fonctions les plus exigeantes en termes de calcul sont celles qui exécutent les règles de vote pour produire des classements des candidats. Par conséquent, on fait des observations sur la fonction `Election.apply_voting_rule` dans le module `electoral_systems.election` avec toutes les extensions désactivées. Les tests on été effectuées lorsque l'application est lancé, i.e. avec la partie graphique, donc on prend aussi en compte la consommation d'énergie de GPU.

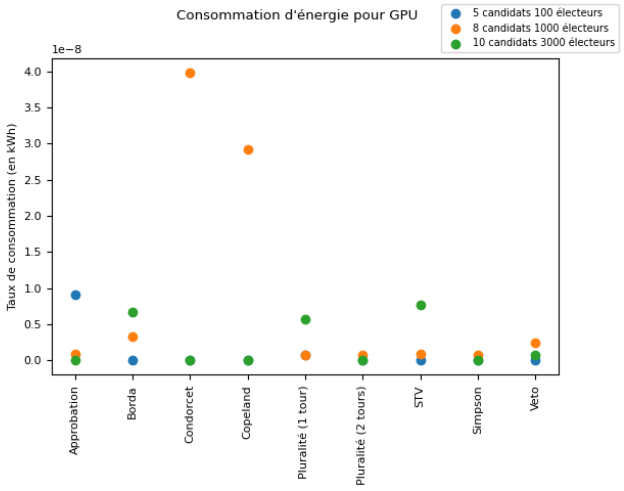
À l'aide de fichier *.csv* généré, on a visualisé les données



On constate que la consommation d'énergie de CPU est globalement similaire pour toutes les règles de vote. Cependant, on remarque que la consommation d'énergie par un algorithme de la règle de vote *Éliminations successives (STV)*. Cela est dû aux plusieurs parcours tous les électeurs (un parcours par un tour). Si on note C le nombre des candidats, N le nombre d'électeurs, alors la complexité $O(N * (C - 1))$, pour les autres règles de vote on a la complexité $\Theta(N)$, sauf la *Pluralité à 2 tours* qui est de complexité $O(N)$. En raison de cette différence de constantes, l'algorithme *STV* présente un pic de consommation d'énergie.

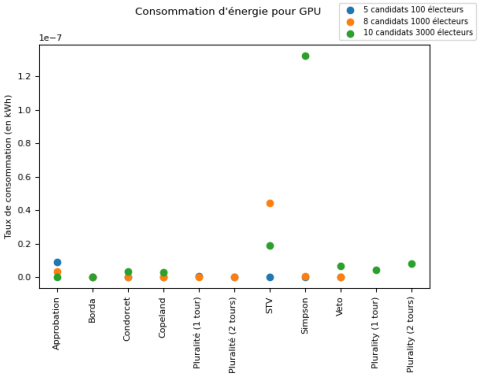


On note que la consommation d'énergie pour RAM est aussi globalement le même sauf le pic pour la règle de vote *STV*. Cela est dû au fait qu'on stocke plus des données, plus précisément, on stocke un classement pour chaque tour. Donc, l'utilisation de mémoire plus élevée par rapport aux autres règles de vote.

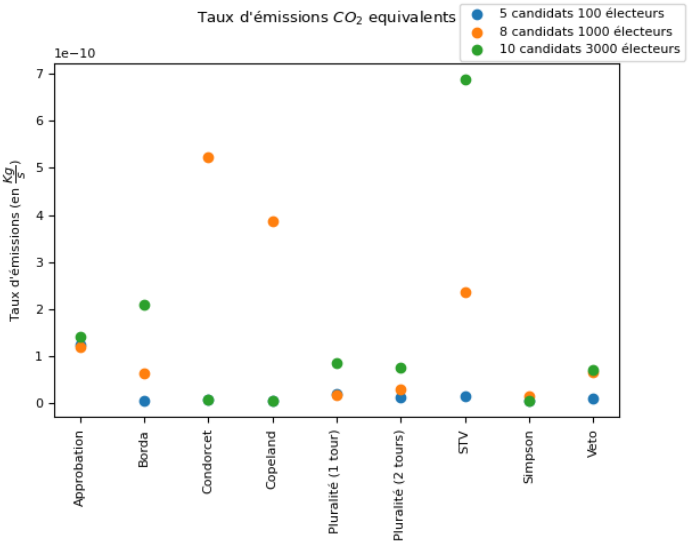


On voit que la consommation d'énergie de GPU est globalement le même sauf les pics éventuels. On pense que c'est le résultat de l'utilisation de *PyQt*.

Il faut remarquer que chaque expérimentation avec les mêmes données produit les pics dans les endroits différents. À droite on présente le graphique de l'autre expérimentation. On voit que les pics apparaissent dans les zones différentes.



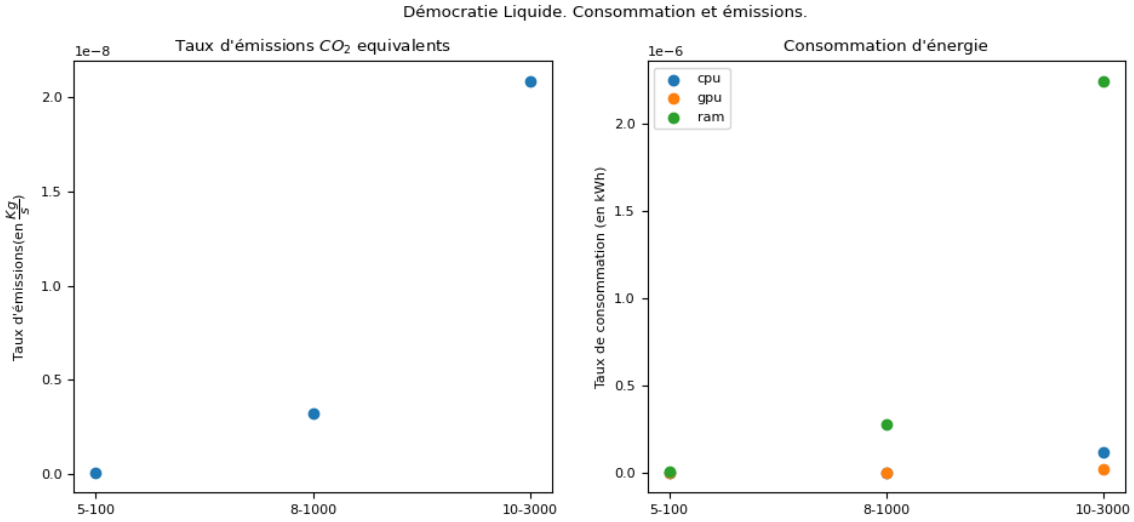
Consommation d'énergie de GPU lors de expérimentation précédente.

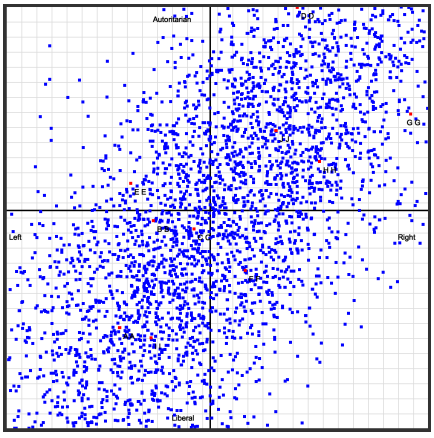


En sachant la consommation d'énergie par notre application, on présente le taux d'émissions CO_2.

De plus, la fonction qui est aussi lourde au niveau de calcul est celle qui effectue les délégations pour la démocratie liquide. Les tests sont effectués sur la fonction `Election._make_delegations` dans le

module `electoral_systems.election`.





Carte politique utilisée pour le test avec 10 candidats, 3000 électeurs

On constate que la consommation de RAM augmente rapidement avec le nombre d'électeurs. Cela est dû au fait qu'on stocke la liste intermédiaire des électeurs (les délégataires possibles) ainsi que les données générées. Comme les électeurs sont très proches les uns des autres, plus le nombre de délégués électeurs stockés est important, ce qui entraîne une augmentation de la consommation de RAM.