Weak Schur numbers

P05 - Formation à la recherche

Romain Ageron, Paul Castéras, Thibaut Pellerin, Yann Portella

1 février 2021

CentraleSupélec

En 1917, le russe **Issai Schur** pose le problème suivant :

En 1917, le russe **Issai Schur** pose le problème suivant :

- On se donne $n \geq 1$ un entier
- $k \geq 1$ un autre entier, qui correspond au nombre de **couleurs**

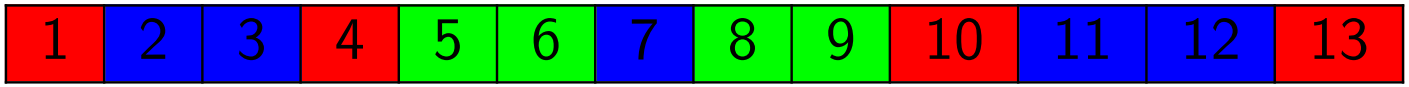
En 1917, le russe **Issai Schur** pose le problème suivant :

- On se donne $n \geq 1$ un entier
- $k \geq 1$ un autre entier, qui correspond au nombre de **couleurs**

Question

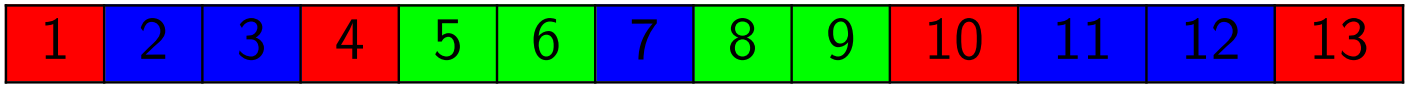
Peut-on colorier les entiers de 1 à n de sorte que si deux nombres ont la même couleur, leur somme n'est pas de cette couleur ? Si oui, un tel coloriage est dit **sans sommes**.

Pour $n = 13$ et $k = 3$, le coloriage



vérifie cette propriété.

Pour $n = 13$ et $k = 3$, le coloriage

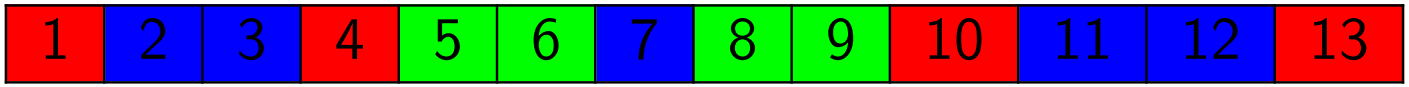


vérifie cette propriété.

Définition

Pour k couleurs, on note $S(k)$ le plus grand entier n tel qu'on puisse colorier les entiers de 1 à n en vérifiant cette propriété. C'est le k-ième **nombre de Schur**.

Pour $n = 13$ et $k = 3$, le coloriage



vérifie cette propriété.

Définition

Pour k couleurs, on note $S(k)$ le plus grand entier n tel qu'on puisse colorier les entiers de 1 à n en vérifiant cette propriété. C'est le k-ième **nombre de Schur**.

Sur l'exemple, on peut vérifier que $S(3) = 13$: on ne peut colorier $[\![1, 14]\!]$ avec trois couleurs.

Définition

Un coloriage est dit **faiblement sans sommes** lorsque pour deux nombres **différents** de même couleur, leur somme n'est pas de la même couleur. On définit de même $WS(k)$ le plus grand entier n tel qu'on puisse colorier les entiers de 1 à n en vérifiant cette propriété plus faible.

Définition

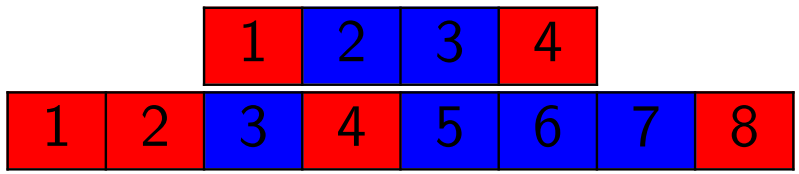
Un coloriage est dit **faiblement sans sommes** lorsque pour deux nombres **différents** de même couleur, leur somme n'est pas de la même couleur. On définit de même $WS(k)$ le plus grand entier n tel qu'on puisse colorier les entiers de 1 à n en vérifiant cette propriété plus faible.

Un coloriage sans sommes et en particulier faiblement sans somme, donc on a toujours $WS(k) \geq S(k)$.

Définition

Un coloriage est dit **faiblement sans sommes** lorsque pour deux nombres **différents** de même couleur, leur somme n'est pas de la même couleur. On définit de même $WS(k)$ le plus grand entier n tel qu'on puisse colorier les entiers de 1 à n en vérifiant cette propriété plus faible.

Un coloriage sans sommes et en particulier faiblement sans somme, donc on a toujours $WS(k) \geq S(k)$.



$S(2) = 4$ mais $WS(2) = 8$

- Pour montrer que $S(k) = n$, il faut trouver un coloriage sans sommes de $[\![1, n]\!]$ à k couleurs **et** montrer qu'on ne peut pas colorier $[\![1, n+1]\!]$.

- Pour montrer que $S(k) = n$, il faut trouver un coloriage sans sommes de $[\![1, n]\!]$ à k couleurs **et** montrer qu'on ne peut pas colorier $[\![1, n+1]\!]$.
- En pratique, on se contente de **minorer** $S(k$. Si on exhibe un coloriage à k couleurs de $[\![1, n]\!]$, on a montré que $S(k) \geq n$.

- Pour montrer que $S(k) = n$, il faut trouver un coloriage sans sommes de $[\![1, n]\!]$ à k couleurs **et** montrer qu'on ne peut pas colorier $[\![1, n+1]\!]$.
- En pratique, on se contente de **minorer** $S(k$. Si on exhibe un coloriage à k couleurs de $[\![1, n]\!]$, on a montré que $S(k) \geq n$.
- Comment effectuer cette minoration ? On peut démontrer des inégalités récursives ou bien rechercher des coloriages par ordinateur.

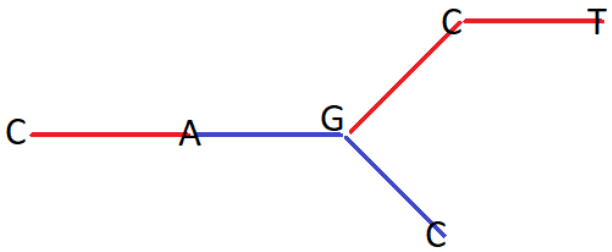
Les recherches récentes sur le sujet se focalisent sur les méthodes numériques.

- On fixe k et on essaye de colorier le plus loin possible
- Le problème peut s'encoder comme une exploration d'arbre, mais le nombre de coloriages possibles explose très vite
- Plusieurs articles récents ont recourt à l'algorithme **Monte-Carlo Tree Search**, et ne considèrent que certains type de coloriages
- On peut également encoder le problème avec un solveur **SAT**

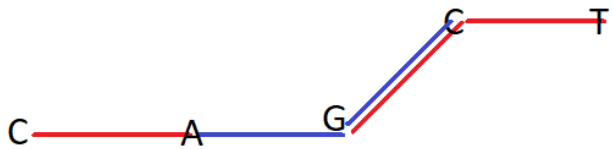
Les recherches récentes sur le sujet se focalisent sur les méthodes numériques.

- On fixe k et on essaye de colorier le plus loin possible
- Le problème peut s'encoder comme une exploration d'arbre, mais le nombre de coloriages possibles explose très vite
- Plusieurs articles récents ont recourt à l'algorithme **Monte-Carlo Tree Search**, et ne considèrent que certains type de coloriages
- On peut également encoder le problème avec un solveur **SAT**

Ces méthodes ont permis d'améliorer les bornes inférieures pour $k \geq 5$ mais peuvent prendre du temps : récemment, le calcul exact de $S(5)$ via un solveur SAT a demandé 20 années de calcul machine.



(a) Bonne distance mais mauvaise orientation



(b) Bonne distance et bonne orientation

Jonction pour une chaîne $GCT...CA$

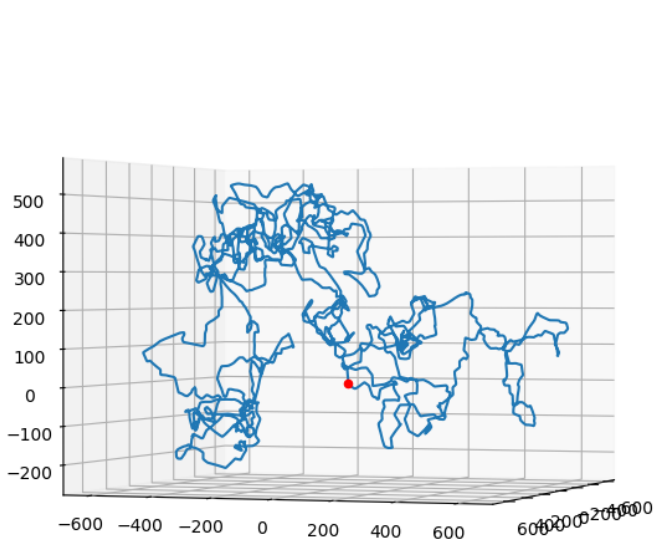
Pour contrôler la qualité de la trajectoire, on ajoute (en bleu) les deux premiers nucléotides à la fin de la chaîne. Ici, G, le premier nucléotide, est bien placé mais l'orientation de GC n'est pas forcément valide.

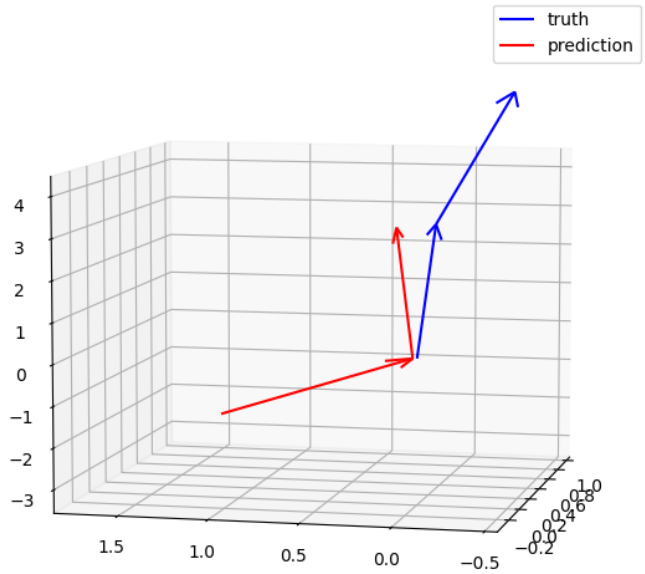
L'étape de mutation s'est révélée très importante :

- elle permet d'explorer l'espace du problème
- des mutations trop limitées bloquent l'algorithme dans des minimums locaux
- des mutations excessives risquent de gâcher des individus prometteurs

Nous avons choisi de faire muter une part importante de la population, mais pas les individus ayant les meilleurs scores. La mutation modifie une des variables, aléatoirement dans l'intervalle associé.

La trajectoire totale est représentée à gauche, et la jonction à droite (un vecteur = un dinucléotide).

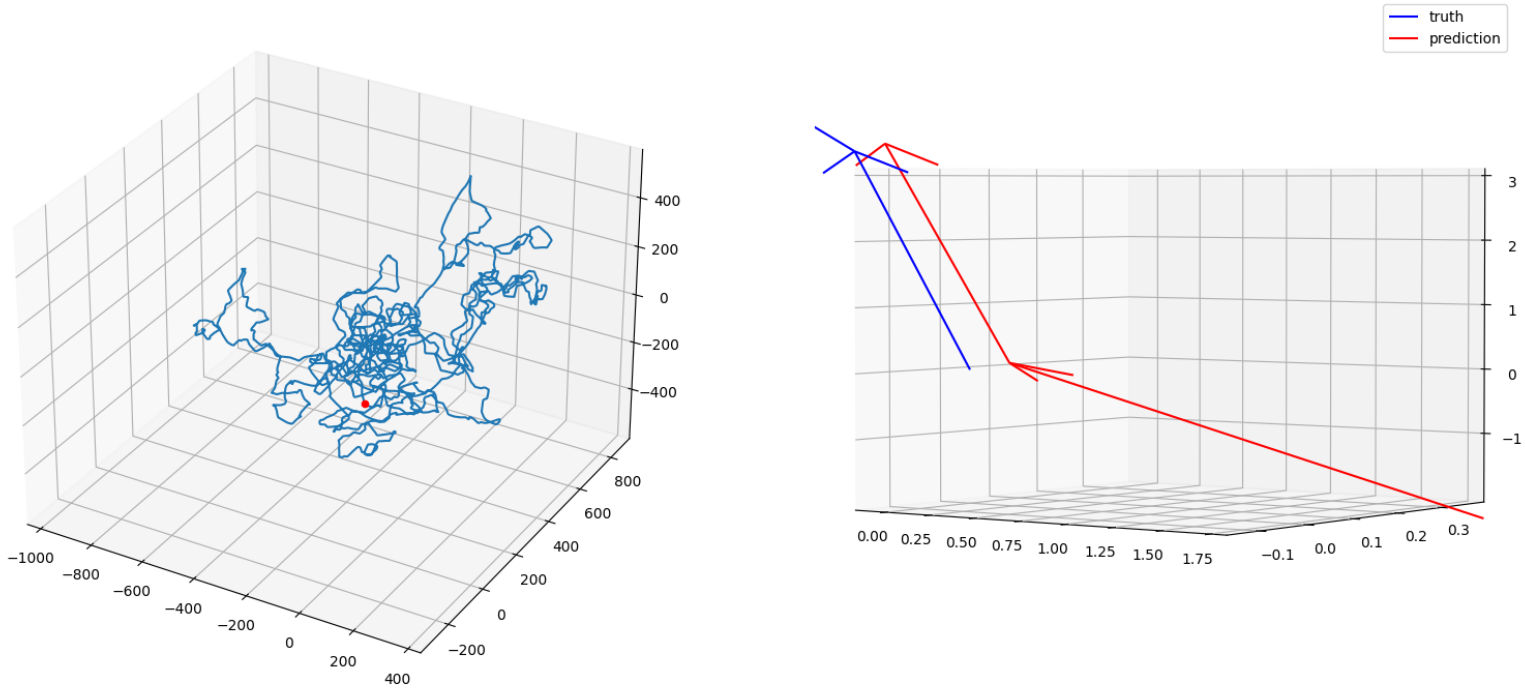




Une chaîne de longueur 8000, avec 200 générations et une population de 1000

Ici, l'erreur de position est faible mais celle d'orientation non négligeable.

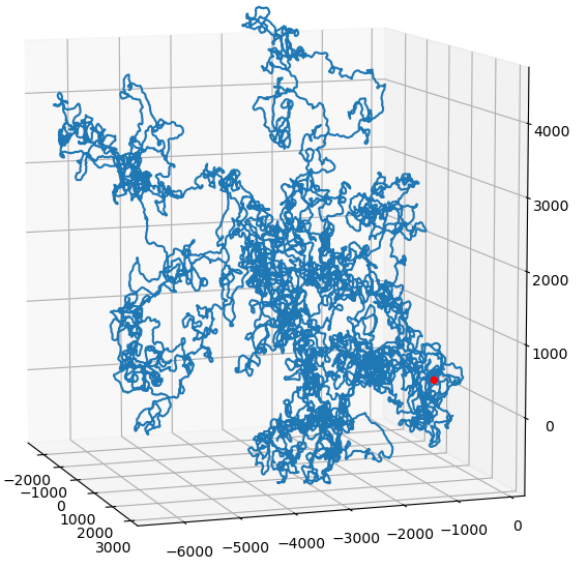
On peut modifier légèrement la fonction afin de pénaliser davantage l'erreur sur l'angle.

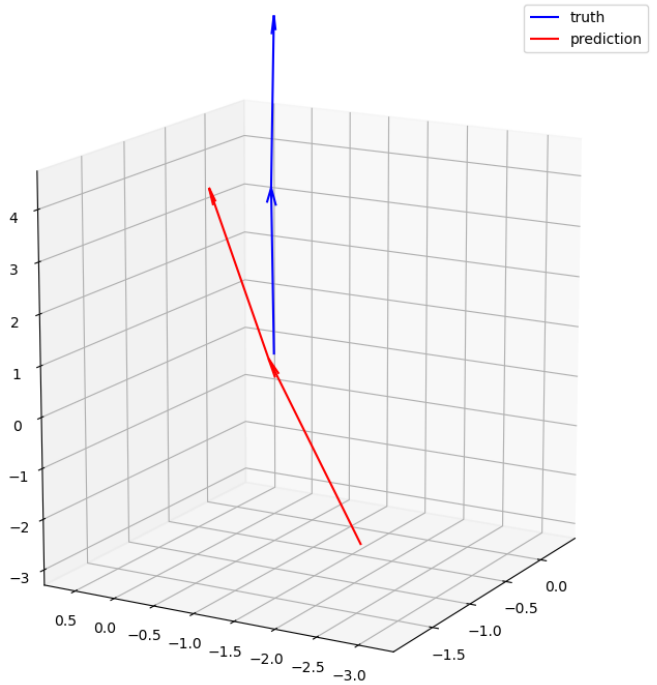


Une chaîne de longueur 8000, avec 250 générations et une population de 200

Les dinucléotides réel et fictif sont ici parallèles, mais la distance est plus importante.

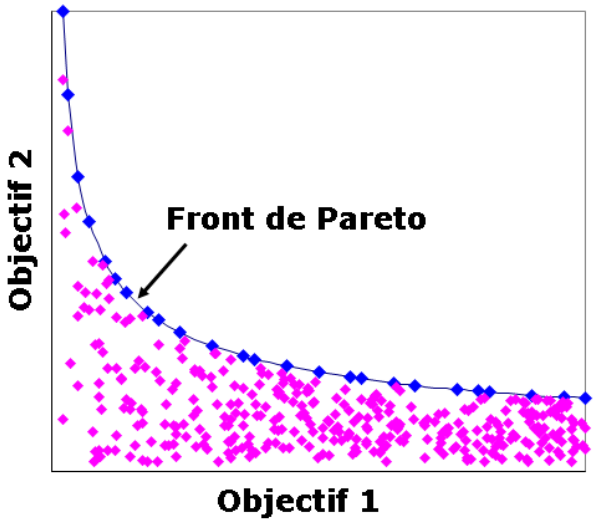
Après une bonne heure de calcul, on obtient ceci pour une chaîne de 180000 nucléotides.





- Le temps d'exécution est d'environ 3 minutes pour une chaîne de longueur 8000, une population de 100 et une centaine de générations
- L'accumulation d'erreurs numériques rend les résultats pour la chaîne de 180k peu fiables
- Difficile d'équilibrer la fonction afin d'avoir une bonne distance et une bonne orientation

Ici, l'optimisation est **multi-critère**. Nous nous sommes ramené à une fonction scalaire, mais il peut être plus efficace de mesurer indépendamment les deux erreurs.
On se prive alors de l'ordre sur $\mathbb{R}$, mais on peut pallier ce problème en utilisant par exemple des *fronts de Pareto*.



Une autre idée serait de faire une fonction de mutation dynamique (par exemple plus de mutation lorsque l'algorithme stagne).