⚙️ Paramètres de la Fonction sigma (ProbCut)

L'objectif de cette étude est de déterminer les paramètres optimaux de la fonction sigma (écart-type) du ProbCut, qui modélise l'incertitude sur la différence de score entre une profondeur donnée D et une profondeur moins élevée d < D.

---

1. Collecte des Données avec Roxane

Nous allons effectuer trois campagnes de collecte de données distinctes, couvrant les phases de jeu (milieu de partie, finale) et une optimisation pour les grandes profondeurs.

A. Configuration TUNE_PROBCUT_MID (Milieu de Partie)

Ce *setting* vise à collecter des données pour les positions de **milieu de partie** (de 8 à 59-D disques).

Protocole de Tirage et d'Analyse

1. **Tirage Aléatoire :**
    - Générer une série de positions de jeu aléatoires, ciblant la zone de **8 à 59-D disques** (discs).
    - Ces positions doivent être légales et jouables.
2. **Analyse de Position :**
    - Pour chaque position tirée, établir une profondeur de recherche maximale, **depth** D.
    - **Référence (Score at D) :** Calculer le score pour cette position à la profondeur **depth D sans appliquer de sélectivité** Noter ce score comme Score_D.
    - **Profondeurs Moins Élevées :** Étudier la même position aux profondeurs **shallow depth d** suivantes, par pas de deux : d in [D&1, D-2].
    - Noter ces scores comme Score}_d.
    - **Couple de Scores :** On obtient ainsi une série de couples de scores (Score_D, Score_d) pour la même position.
3. **Classification et Enregistrement :**
    - Classer chaque couple de scores par le **nombre de disques** sur le plateau.
    - Écrire une ligne par couple dans un fichier texte nommé probcut_mid.txt (et ses extensions) au format :
      "n_discs", "depth", "shallow depth", "Score_D - Score_d)
    - **Plan d'Itérations (Critère de Temps "Raisonable")**

Profondeur D (depth)	Nombre d'Itérations	Fichier de Sortie
<= 15	2000 à 4000	probcut_mid.txt
16 <= D <= 17	1000 à 2000	probcut_mid_ext1.txt
18 <= D <= 19	500 à 1000	probcut_mid_ext2.txt
...	...	...

-

B. Configuration TUNE_PROBCUT_END (Finale)

Ce *setting* réplique le protocole ci-dessus mais se concentre sur les positions de **finale** .

Protocole et Enregistrement

- Le protocole de tirage, d'analyse et d'enregistrement des couples de scores est **identique** à celui de TUNE_PROBCUT_MID, mais adapté aux positions de fin de partie.
- L'enregistrement légèrement different se fera dans **probcut_end.txt** (et ses extensions).

Plan d'Itérations

Profondeur D (depth)	Nombre d'Itérations	Fichier de Sortie
<= 25	2000 à 4000	probcut_end.txt
26 <= D <= 27	1000 à 2000	probcut_end_ext1.txt
28 <= D <= 29	500 à 1000	probcut_end_ext2.txt
...	...	...

C. Configuration TUNE_PROBCUT_END2 (Finale/Grandes Profondeurs)

Ce *setting* vise à obtenir des données pour les très grandes profondeurs en tirant parti des résultats d'analyse déjà effectués.

Optimisation pour Grandes Profondeurs

- **Utilisation de Résultats Pré-Calculés :** Au lieu d'effectuer une nouvelle recherche complète pour Score_D, réutiliser des positions dont l'évaluation à grande profondeur est déjà stockée ou a été calculée.
- **Objectif :** Gain de temps substantiel en évitant le calcul coûteux des Score_D pour les plus grandes profondeurs, tout en calculant Score_d plus rapides.
- Le format de sortie reste le même.

---

2. Détermination des paramètres

La détermination des paramètres repose sur l'analyse statistique des données collectées ainsi que sur l'ajustement d'un modèle décrivant l'évolution des évaluations au cours de la partie.

1. Lancement du script de modélisation

Après avoir collecté l'ensemble des données nécessaires, il convient d'exécuter l'un des scripts suivants :

- `model_2zones.py`
  Ce script calcule deux séries distinctes de paramètres :
  - une série **midgame**,
  - une série **endgame**.

  Avec un lissage progressif entre les deux zones afin d'assurer une transition cohérente.

- `model_sigma.py`
  Ce script calcule au contraire **une série unique de paramètres**, basée sur un modèle continu.

Dans la pratique, **aucune différence nette** n'a été observée entre les résultats produits par les deux méthodes.

2. Intégration des paramètres

Une fois les paramètres de la fonction sigma déterminés et validés, il faut les **intégrer dans Roxane** *(RXEngine::sigma)*, où ils seront utilisés par le moteur pour ajuster dynamiquement son comportement en fonction du stade de la partie.