Théorie des réseaux complexes, réseaux dirigés et processus de décision: le cas du jeu de go

Écrit par: Célestin Coquidé[1] le 23/06/16

Stage effectué au Laboratoire de Physique Théorique de Toulouse

Encadrants:

Bertrand Georgeot[2] Olivier Giraud[3]

1 Université de Montpellier, Master 2 Physique et Ingénierie du Vivant.

2 Laboratoire de Physique Théorique, Université Paul Sabatier Toulouse.

3 Laboratoire de Physique Théorique et Modèles Statistiques, Université Paris sud Saclay.

Première partie

Introduction

Nous allons étudier un jeu de stratégie datant du VIII^{ème} siècle de notre ère, le go. Ce jeu se joue sur un plateau appelé *goban* composé de lignes verticales et horizontales. Chaque joueur place une pierre de couleur noire ou blanche à son tour de jeu. Le but du jeu est de créer de larges territoires protégés et de capturer les pierres adverses. Le go a déjà été étudié en tant que réseau complexe, pour ce qui est des parties humaines et pour plusieurs niveaux, professionnels et amateurs, pour en tirer des modèles locaux de coups stratégiques et contribuer à l'amélioration des simulateurs de jeux de plateau[1]. Cependant dans un monde qui change sur le plan technologique et plus particulièrement informatique avec l'avènement de l'intelligence artificielle, il est très intéressant de compléter les précédentes études sur le go avec des parties faites par ordinateur dans l'optique de quantifier les différences entre hommes et machines dans la prise de décision stratégique et pourquoi pas d'aboutir à un test de Turing pour le go.

Le terme *"graphe"* $G = (E, V)$ est apparu pour désigner un ensemble G de nœuds notés V connectés par des liens notés E. Les mathématiciens et plus précisément les praticiens de la topologie combinatoire ont su étudier ce type d'objet topologique de diverses façons et trouver de nombreuses applications.

Il faut attendre la fin des années 90 pour que les physiciens s'approprient se sujet d'étude avec le terme *"réseau"*. Une des applications les plus connues est le World Wide Web et l'algorithme de recherche appelé PageRank, la base du moteur de recherche *GOOGLE*, élaboré par Larry PAGE et Sergey BRIN fondateurs de *GOOGLE*.

La difficulté de cette tâche est que la matrice représentant les nœuds (page web) et les liens (hyperliens entres les pages) du World Wide Web est de taille $N \times N$ avec N de plus en plus grand au fil des années. Par exemple aujourd'hui le World Wide Web est un réseau dirigé comportant 47.5 milliards de pages indexées sur Google [2].La matrice carrée A où A_{ij} est différent de 0 si et seulement si une page j pointe vers i, cette matrice d'hyperliens après normalisation et transformation devient la **matrices de Google**. Le PageRank est ni plus ni moins qu'un vecteur propre de cette matrices de Google dont l'entrée i est assimilable au temps que passerait un surfeur web aléatoire sur la page i, c'est donc une information sur l'importance de cette page. [3]

Pour ce stage effectué au sein de mon laboratoire d'accueil, j'ai pu étudier des articles récents sur les réseaux. Il y a différents types de réseaux : réels, aléatoires, complexes, dirigés non dirigés, pondérés, non pondérés et dynamiques.

Deuxième partie

Bibliographie et état de l'art

Un jeu de plateau vue comme un réseau complexe dirigé :

Un réseau étant un ensemble de nœuds reliés les uns aux autres par des liens, il est possible de construire un tel objet avec un jeu de plateau. Dans le cas du go, la stratégie est uniquement positionnelle car toutes les pierres jouées par les deux joueurs ont même valeurs, il n 'y a ni roi, ni dame, ni pions comme pour les échecs, de plus le goban qui est le plateau de go est carré et avec des lignes horizontales et verticales. On peut identifier un nœud comme étant un schéma de jeu local défini par exemple par un carré de 3×3 intersections et si on joue une plaquette "j" proche d'une autre plaquette "i" il y a alors un lien dirigé allant de "i" vers "j". On peut alors construire un tel réseau avec différents types de banques de données de parties de go, comme par exemple avec des parties amateurs et professionnelles[1]. On peut aussi construire un réseau avec le go en prenant d'autres types de plaquettes, une étude visant à construire d'autres réseaux avec deux autres types de plaquettes pour le go a aussi étudié les différentes phases de jeu : le début, le milieu et la fin de partie[9].

La statistique des liens dans chaque réseau est représentée par la distribution intégrée des liens et elle nous donne des informations sur la connectivité du réseau. Si cette distribution suit une loi de puissance,

nous avons un réseau invariant d'échelle, c'est le cas pour les réseaux construits à partir de parties de go, de plus la distribution intégrée des liens entrants est proche de celle des liens sortants [1], [9], en revanche des réseaux comme le web ont une distribution asymétrique entre liens entrants et sortants. Les distributions intégrées de liens pour les réseaux réels suivent souvent une loi de puissance [13].

On peut récupérer d'une banque de données la matrice des liens entre chaque nœud du réseau. La matrice est diagonalisable, numériquement grâce à une bibliothèque *LAPACK* (Fortran/C/C++) par exemple, et on peut en extraire des valeurs propres et des vecteurs propres. Notons que la non-symétrie de cette matrice donne lieu à deux ensembles de vecteurs propres : les droits et les gauches. Le premier vecteur propre droit est celui qui est utilisé dans l'algorithme du PageRank pour le moteur de recherche *Google*. Il est aussi nommé le Perron Vecteur, nom qui vient du théorème de Perron-Frobenius (1912). Initialement pensé par Perron pour des matrices strictement positive et sans entrées nulles, il a été reformulé par Frobenius pour toutes entrées positives et nulles[3].

Afin de comparer plusieurs réseaux entre eux, le spectre des valeurs propres de la matrice associée à chaque réseau est utilisé. Il y a un gap plus ou moins important entres les plus grandes valeurs propres et les valeurs propres du *"bulk"* et ce gap diffère d'un réseau à l'autre, dans le cas du go le gap change quand on change de type de nœud[9]. Le bulk est la zone dense du spectre de valeurs propres.

On peut aussi utiliser une autre grandeur, le rapport de participation inverse ou IPR en anglais qui est utilisé en physique du solide afin de quantifier la localisation d'un vecteur propre. Cet outil permet d'analyser la structure des vecteurs propres associés a un réseau et a été utilisé pour différents réseaux complexes dirigés[17].

De nombreuses autres applications :

Les biologistes utilisent de plus en plus les réseaux d'interactions protéines-protéines et ces réseaux permettent de voir directement les communautés de protéine constituants divers complexes biochimiques. Grâce aux réseaux construits à partir de plusieurs banques de donnée de séquence ADN pour différentes espèces vivantes, nous pouvons repérer les différences génétiques entre deux individus ou bien construire un arbre phylogénétique. On peut construire des réseaux d'ADN pour différentes longueurs de mot, où un mot est une suite de bases azotés A,T,G ou C de longueur m ,un lien joint deux mots se suivant sur une même séquence ADN[4].

La loi de Zipf est une loi de distribution suivant une loi de puissance et décrit par exemple la distribution intégrée des fréquences d'utilisations des mots dans la langue anglaise [5], des tailles des villes [6] et des ouvertures aux échecs [7]. Elle est présente dans les réseaux biologiques et aussi dans nos réseaux de go.

Ce qui n'a pas encore été fait avec le jeu de go, c'est l'élaboration de réseaux pour les parties jouées par des ordinateurs.

Un contexte très intéressant : Jouer est quelque chose d'ancré en nous, par là je veux dire que n'importe quel être humain depuis des milliers d'années, a au moins une fois joué seul ou avec un autre être humain à un jeu. Dans un jeu, il y a des règles, mais aussi il y a une notion de coup *"stratégique"* dans les jeux comme les échecs où encore le go. En plus d'avoir accès implicitement au processus de décision du cerveau humain, grâce à l'étude d'un jeu de plateau comme le go, nous avons la possibilité de voir les différences majeurs entre l'humain et l'ordinateur dans la façon de jouer à un jeu de plateau, ici le go.

Dans ce rapport je présenterai de nouveaux éléments et nous pourrons alors discuter de la différence entre humains et ordinateurs dans la stratégie d'un jeu commun, le go. J'essaierai d'aller vers une quantification de cette différence avec l'élaboration d'un **test de Turing pour le go**. Ce test mettra en pratique divers outils utilisés dans ce rapport afin d'établir si une banque de donnée est humaine ou bien simulé par un ordinateur. Il faut savoir que le test de Turing est un test proposé par Alan Turing en 1950 afin de constater si une intelligence artificielle peut imiter la conversation humaine[8]. Dans notre cas si on ne peut pas distinguer l'ordinateur de l'humain avec ce test on peut dire que l'ordinateur imite parfaitement un joueur de go humain, ainsi ce test pourrait être utile pour par exemple justifier le niveau professionnel d'un simulateur comme AlphaGo.

Troisième partie

Méthodes

Ici je vais présenter les différentes méthodes utilisées au cours de mon stage, il s'agira d'une liste explicatives des différentes étapes qui m'ont permis de recueillir les résultats qui vous seront présentés dans ce rapport.

Une banque de donnée de parties de jeu se constitue d'un ensemble de fichier texte au format ".sgf" pour Smart Game Format, on y retrouve alors des informations relatives à une partie comme par exemple les noms des deux joueurs, leurs niveaux exprimés en *dans* de 1 à 9 pour les professionnels et suivi d'une autre suite de 1 à 9 pour les amateurs. On peut savoir le type de tournoi, le type de règle utilisé, les scores finaux et surtout nous y trouvons une suite de positions de pierres jouées par chaque joueur voir fig.20 en annexe.
Mon travail a consisté à élaborer et étudier la première banque de données relative à des parties jouées par ordinateur pour différentes tailles de goban 19×19 et 9×9, pour cela j'ai dû utiliser un simulateur de go.

Avant 2016 aucun des simulateurs de go n'a réussi à battre un très bon joueur sans quelques handicaps majeurs. Parmi les simulateurs il y a deux familles : les déterministes et les algorithmes de type Monte-Carlo. Un algorithme déterministe à la différence d'un type Monte-Carlo, joue toujours la même partie si les conditions initiales sont identiques en revanche dans le cas d'un algorithme Monte-Carlo chaque coup est joué aléatoirement et testé dans plusieurs parties créées en parallèles afin d'attribuer une valeur au coup et ainsi même avec des conditions initiales identiques l'ordinateur simulera des parties différentes. Cependant en mars 2016 un simulateur du nom d'AlphaGo à enfin été victorieux, le 15 mars 2016 Lee Sedol numéro 3 mondial perd 4-1. AlphaGo couple le Monte-Carlo avec le deep learning (réseaux neuronaux artificiels).

Le plus simple d'accès et le plus simple à automatiser des simulateurs que nous avons pu trouver, est Gnugo. Gnugo 3.8 est un programme libre que vous pouvez télécharger ici [10]. La commande "./gnugo - -help" affiche le manuel d'utilisation, il est aussi accessible sur ce site [11].

Afin de construire notre banque de données on a tout d'abord lancé des milliers de parties à la suite de l'ordinateur contre lui-même. Nous avons ensuite comparé le réseau construit à partir de ces parties avec ceux construits à partir de parties amateurs, dont les banques de données ont étés prises sur ce site [12].

1 Acquisition des données

Gnugo, bien que déterministe, effectue son premier mouvement grâce à un nombre appelé *"graine"*. Nous devons avoir un ensemble de parties successives totalement indépendantes car il est important que les parties le soient si nous voulons voir comment l'ordinateur joue, si les parties se ressemblent trop ou si elles sont identiques la matrice des liens de notre réseau aura des entrées surpondérées sans raison stratégique.

Après avoir simulé environ 10 000 parties, nous nous sommes rendus compte qu'il y avait beaucoup de ressemblance entre les fichiers sgf obtenus. Nous avons alors décidé de construire nous même des listes de nombres pris aléatoirement entre 1.10^9 et 2.10^9 avec un petit programme en C et la graine de départ du simulateur sera ce nombre.
Pour le format 9×9, nous avons fait deux types de parties, les parties déterministes, donc avec un algorithme identique à celui utilisé pour Gnugo 19×19 et les parties Monte-Carlo grâce à l'option Monte-Carlo du simulateur, qui ne fonctionne que pour des tailles de goban inférieur ou égale à 9×9. Par défaut, cette option simule 80 000 parties par coups joués[11].

2 Construction du réseau

Un réseau est un ensemble de nœud interconnectés les uns aux autres, dans le cas d'un réseau dirigé ces liens ont une direction. Soit un coup joué en position (h, v) du goban, avec h et $v \leq 19$ ou pour l'autre format $h, v \leq 9$, alors on relie un coup à un autre si et seulement si il existe un coup joué précédemment en position

$(h \pm d_s, v \pm d_s)$. Les premières études ont montrées que selon la valeur de cette distance d_s, les réseaux sont différents et il y a eu des résultats pour plusieur d_s[1]. Dans notre étude, on choisira $d_s = 4$.

Le nœud du réseau est une plaquette, représentant une bataille locale sur le goban, c'est un carré de 3 par 3 intersections et la convention est la suivante : le centre du carré est la position de la pierre qui vient d'être joué. Les plaquettes doivent être non équivalentes c'est à dire que par symétrie de rotation, d'axe et par échange de couleurs elles sont toutes distinctes les unes des autres. Nous avons alors 1107 plaquettes non équivalentes pour ce type de réseau. Par convention encore, nous choisirons de représenter les plaquettes avec comme couleur de la pierre centrale, la couleur noire. Une illustration pour comprendre cette notion de non équivalence voir fig.1. D'autres tailles et formes de plaquettes sont bien évidement possible et le plus grand réseau étudié est constitué de 193995 nœuds[9]. La suite de ce travail est faite avec un réseau de $N_v = 1107$ nœuds.

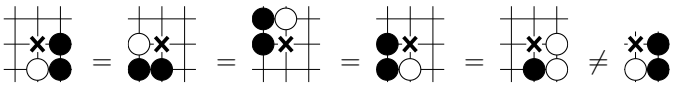


FIG. 1 – Différentes plaquettes équivalentes et non équivalentes, de gauche à droite : une plaquette "i" puis une plaquette équivalente à "i" par rotation de $\frac{\pi}{2}$ ensuite une autre plaquette équivalente à "i" par rotation de π ensuite une autre plaquette équivalente à "i" par symétrie d'axe y ensuite une dernière plaquette équivalente à "i" si blanc joue au milieu et enfin une plaquette non équivalente qui correspond à un coin du goban.

3 Traitement des données

Je me suis inspiré d'un programme écrit en C++ par un ancien thésard du laboratoire où j'ai effectué mon stage, il récupère depuis un ensemble de plusieurs fichiers sgf, la liste des coups joués pour chacune des parties. Ce programme garde en mémoire les positions x et y de chaque pierre sur le goban, leur couleur et retire les pierres capturées lors de la partie.

Nous obtenons un fichier "matrix.out" indiquant le nombre de nœuds du réseau utilisé, ici 1107, le nombre de liens avec dégénérescence et la liste de lignes indiquant le nœud source et le nœud cible de chaque lien.
J'ai écrit un programme afin de construire la matrice d'hyperliens 1107×1107 suivant cet algorithme :

$$H_{ij} = \begin{cases} 1/l_j & \text{si } P_j \in B_i \\ 0 & \text{sinon.} \end{cases} \quad (1)$$

Cet algorithme vient du PageRank de Google et P_j dénote la page j et B_i est l'ensemble des pages j pointant vers la page i. l_j est le nombre de liens total sortant de la page j. $\sum_{i=1}^{N_v} H_{ij} = 1$ si au moins une page pointe vers "i". Cette matrice représente une chaîne d'évènements markoviens et est donc stochastique par colonne.
Principe de l'algorithme du PageRank :

prenons un surfeur web aléatoire qui parcourt toutes les pages aléatoirement sur le web et π un vecteur de probabilités de présence de ce surfeur sur chaque nœud du réseau web, faire l'application $A\pi$ une fois revient à faire avancer d'un pas le surfeur dans le réseau et si on applique une infinité de fois on converge vers la distribution asymptotique des temps que passe le surfeur sur chaque site web.

S est la matrice stochastique construite à partir de H tel que :

$$S_{ij} = \begin{cases} \frac{1}{N_v} & \text{si pour j fixé } \forall \text{ i } H_{ij} = 0 \\ H_{ij} & \text{sinon.} \end{cases} \quad (2)$$

S permet de supprimer les *"dangling nodes"*, c'est des nœuds qui ne pointent vers aucun autre nœud du réseau. Si on parcourait le réseau en partant de n'importe quel nœud et en suivant les liens, si on arrive à un dangling node, nous y sommes bloqués indéfiniment, dans le cas du web si on arrive sur une page internet dans laquelle il n'y a aucun lien cliquable, on est sur un dangling node et dans ce cas il suffit de taper n'importe quelle adresse web pour en sortir, c'est de cette manière qu'agit S pour qu'un dangling

node devienne un nœud connécté à tous les autres. On remplace alors les entrées correspondantes par $\frac{1}{N_v}$ comme dans (2). Enfin G la matrice final dite matrices de Google :

$$G_{ij} = \alpha S_{ij} + (1-\alpha)\frac{1}{N_v} \tag{3}$$

Où α est le *"damping factor"*, il permet de supprimer le problème des *"dangling groups"*. Ce facteur est très important car le PageRank dépend de la valeur de α, en effet si il est proche de 1 la méthode de puissance permettant d'obtenir le PageRank va mettre beaucoup plus de temps à converger que pour une plus petite valeur de α si la première valeur propre est dégénérée[3]. Avec une valeur de α de 0.85, λ_1 n'est plus dégénérée. Sergey BRIN et Larry PAGE ont choisis $\alpha = 0.85$ et nous ferons de même.

Une fois cette matrice construite, nous utilisons *LAPACK* et la sous-routine DGEEV pour la diagonaliser. Le premier vecteur propres droit ψ_1 est tel que $G\psi_1 = \lambda_1\psi_1$ et toutes ses composantes ont même signe et la valeur propre associé est $\lambda_1 = 1$. Ce vecteur est le PageRank est sa $i^{ème}$ composante correspond a la probabilité qu'un surfeur aléatoire reste sur le nœud i du réseau. Dans notre cas un nœud n'est pas une page web mais un motif de jeu local et le PageRank nous renseigne sur les coups les plus importants d'une banque de données.

Nous pouvons aussi étudier le réseau construit en échangeant liens entrants et liens sortants la matrices de Google associée est notée G^* et on appelle le PageRank associé le CheiRank, qui donne une information supplémentaire sur le réseau.

Quatrième partie
Résultats

A partir des méthodes listées dans la partie précédente de ce rapport, nous avons obtenu plusieurs résultats intéressant sur les réseaux complexes dirigés construits avec des banques de données de parties d'humains amateurs sur un goban de taille 19×19, obtenues depuis U-go, et de parties simulées par ordinateur avec le programme Gnugo sur une même taille de goban. Nous avons aussi des résultats pour des réseaux construits à partir de parties simulées par ordinateur pour une taille de plateau de jeu plus petite (9×9) et pour deux types d'algorithmes : le Monte-Carlo et le non Monte-Carlo (déterministe).

4 Distributions intégrées liens entrants/sortants

Nous voulons tracer la probabilité qu'un nœud du réseau ait au plus k liens avec ces proches voisins en fonction de k, pour tous les réseaux que nous avons construits.

La distribution intégrée des liens entrants et sortants pour les réseaux construits à partir de 4000 parties U-go (humain) et 4000 Gnugo (ordinateur) pour un goban 19×19 est montrée sur la fig.2, les pentes sont toutes deux proches de -1, la distribution suit donc une loi de puissance, on parle alors de réseaux invariant d'échelle. $\bar{k}$ le degré moyen est défini ainsi :

$$\bar{k} = \sum_{i=1}^{N_v} \frac{k_i}{N_v} \tag{4}$$

Un liens entrant dans le nœud V_j depuis V_i est un liens sortant de V_i allant à V_j, donc tout nœud du réseau connecté est à la fois une sortie et ou une entrée donc $\bar{k}_{entrants} = \bar{k}_{sortants}$ on notera alors $\bar{k}$.

Les $\bar{k}$ obtenus : $\bar{k} = 693.5$ pour U-go (humain) et $\bar{k} = 715.9$ pour Gnugo (ordinateur).

La symétrie entre liens entrants et liens sortant est aussi présente dans l'étude faite sur le go avec des parties professionnels avec une pente de -1.03[9]. Contrairement à nos réseaux le Web a un exposant de -1.7 pour les liens sortants et -1.1 pour les liens entrants donc une asymétrie entre type de lien[14]. Pour les réseaux construits avec des séquences d'ADN, il y a un exposant moyen valant 5 pour les liens entrants[4]. Une première différence se trouve pour les grands degrés, où on observe un décrochage de la partie linéaire pour les réseaux ordinateurs Gnugo 19×19.

Ensuite nous avons tracés la distribution intégrée des liens entrants/sortants des réseaux construits à partir de 40 000 parties ordinateurs au format 9×9 et pour deux types d'algorithmes : l'un déterministe et l'autre Monte-Carlo. Afin de pouvoir comparer entre taille de goban pour les réseaux ordinateurs, on a pris le même rapport de nombres de parties entre format de goban que celui du nombres de pierres jouées entre une partie en 19×19 et en 9×9 soit un rapport de 5. Afin de comparer les différents algorithmes Gnugo pour différent nombre de partie, nous avons construit deux réseaux de 10 000 parties chacun pour les deux types d'algorithmes noté $^{(g1)}$ et $^{(g2)}$.

On observe une différence de la pente avec les précédents réseaux 19×19, la différence entre types d'algorithmes pour la pente est bien moins forte. En effet on passe de -1 à -0.84 (Gnugo sans Monte-Carlo) et à -0.80 (Gnugo avec Monte-Carlo) comme vous le montre la fig.3. Le décrochage de la partie linéaire est encore présent ici.

En changeant la taille de l'échantillon pour Gnugo 9×9 avec et sans Monte-Carlo, on observe un changement de pente et on peut observer un décalage vers les bas degré pour le décrochage, en effet k_c passe de 3.3-3.7 à 2.9-3.0 voir fig.3 et fig.4.
Les $\bar{k}$ obtenus : $\bar{k} = 637.7$ non Monte-Carlo de 20 000 parties, $\bar{k}^{(g1)} = 320.2$ et $\bar{k}^{(g2)} = 317.5$ groupes non Monte-Carlo de 10 000 parties, $\bar{k} = 638.5$ Monte-Carlo de 20 000 parties et enfin $\bar{k}^{(g1)} = 319.9$ et $\bar{k}^{(g2)} = 318.6$ groupes Monte-Carlo de 10 000 parties.

Les distributions intégrées des liens des réseaux que nous avons construit suivent une loi de puissance dont les pentes sont proche de -1 pour les grands formats et proches de -0.8 pour les petits formats de plateau. Ces exposants sont caractéristique des réseaux invariants d'échelle.

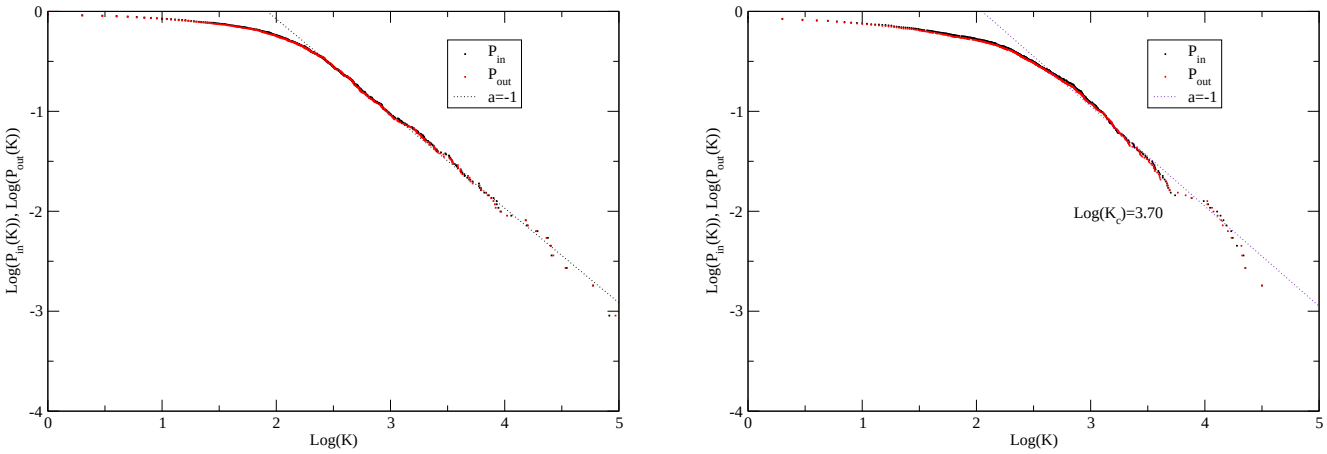


FIG. 2 – Distribution intégrée des liens entrants (P_{in}) en noire et sortants (P_{out}) en rouge pour U-go (humain) 19×19 4000 parties à gauche et Gnugo (ordinateur) 19×19 4000 parties à droite, courbe en pointillé noire est une droite de pente -1 on voit bien que les distributions intégrées des liens entrants et sortants ont même exposant et on a deux réseaux invariants d'échelle. A droite un décrochage en $\log(K) = 3.70$ est bien plus prononcé que dans la figure de gauche.

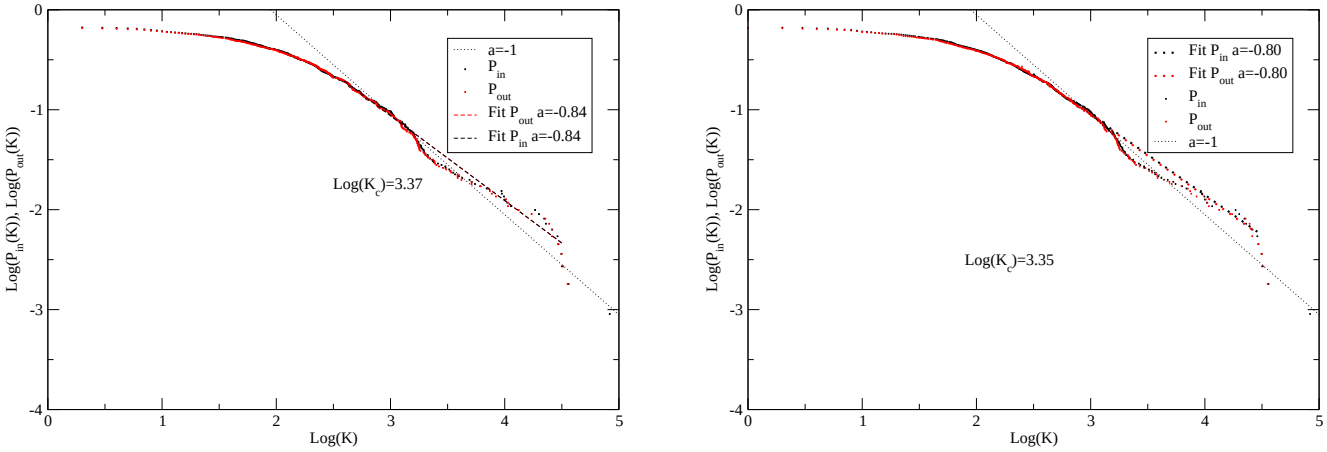


FIG. 3 – Distribution intégrée des liens entrants (P_{in}) en noire et sortants (P_{out}) en rouge pour Gnugo (ordinateur) 9×9 non Monte-Carlo 20 000 parties à gauche et Monte-Carlo 20 000 parties à droite, courbe en pointillé noire est une droite de pente -1 et la courbe en pointillé large noire est le fit pour les liens entrants de pente -0.84 pour non Monte-Carlo et -0.80 pour Monte-Carlo et la droite en pointillé large rouge est le fit pour les liens sortants de pente -0.84 pour non Monte-Carlo et -0.80 pour Monte-Carlo, on voit bien que les distributions intégrées de liens entrants et sortants ont même exposant pour ces deux réseaux et ils sont donc invariants d'échelle. On observe un fort décrochage en $\log(K) = 3.37$ pour non Monte-Carlo et $\log(K) = 3.35$ pour Monte-Carlo.

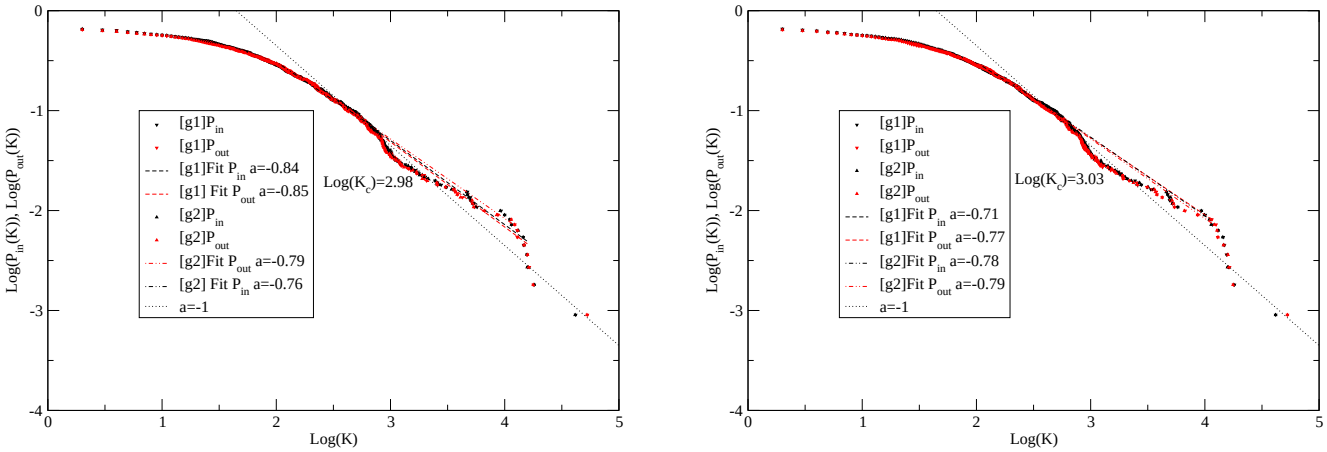


FIG. 4 – Distribution intégrée des liens entrants (P_{in}) en noire et sortants (P_{out}) en rouge Gnugo (ordinateur) 9×9 groupe 1 ([g1]) et groupe 2 ([g2]) non Monte-Carlo de 10 000 parties chacun à gauche et Gnugo 9×9 Monte-Carlo groupe 1 ([1]) et groupe 2 ([2]) de 10 000 parties chacun à droite, courbe en pointillé noire est une droite de pente -1, la courbe en tirets larges noire est le fit pour les liens entrants des groupes 1 de pente -0.84 pour non Monte-Carlo et -0.71 pour Monte-Carlo, la courbe en tirets larges rouge est le fit pour les liens sortants des groupes 2 de pente -0.85 pour non Monte-Carlo et -0.77 pour Monte-Carlo, la courbe en pointillé tirets noire est pour les liens entrants des groupes 2 de pente -0.76 pour non Monte-Carlo et -0.79 pour Monte-Carlo et la courbe en pointillé tirets rouge est pour les liens sortants des groupes 2 de pente -0.79 pour non Monte-Carlo et -0.78 pour Monte-Carlo, on voit bien que les distributions intégrées de liens entrants et sortants ont des exposants assez proches pour quatre réseaux, nous avons des réseaux invariants d'échelle et un décrochement prononcé en $\log(K) = 2.98$ pour non Monte-Carlo et $\log(K) = 3.03$ pour Monte-Carlo.

5 Spectres de valeurs propres des matrices Google associées

Après avoir construit la matrice de Google associée à chaque réseau, nous pouvons grâce à *LAPACK* diagonaliser de telles matrices et ainsi obtenir différents spectres.

Il se distingue deux ensembles de spectres, les spectres ordinateurs sont globalement semblables et se distinguent parfaitement des spectres humains. Les spectres pour G^* sont similaires aux spectre pour G voir les fig.21, 22, 23 et 24 en annexe. La zone dense en valeurs propres et bien plus compacte pour les humains que pour les ordinateurs voir les fig.5 et fig.6 pour 9×9. Une première question que nous nous sommes posés est de savoir si la taille de l'échantillon changerait le spectre Gnugo. Pour ce qui est des humains des résultats sur plusieurs tailles d'échantillons et plusieurs niveaux de joueurs ont montrés que le spectre resté globalement le même[1] et [9]. Pour vérifier la variabilité entre les spectres de différents nombres de données nous avons tracés le spectre des valeurs propres pour plusieurs sous groupes de 10 000 parties Gnugo (ordinateur) pour un goban de taille 9×9 pour les deux d'algorithmes non Monte-Carlo et Monte-Carlo voir fig.6.

Il apparaît que les spectres ordinateurs se distinguent plus des spectres humains que des spectres pour différents nombres de données et des spectres pour différents algorithmes.

Afin de pouvoir avoir une idée de comment les valeurs propres se distribuent dans ces spectres, nous avons tracé le nombre de valeurs propres en échelle log présent dans des anneaux concentriques de surfaces différentes à l'intérieur de la partie dense du spectre voir la fig.7.

$$N(x) = \#\lambda \text{ tel que } (x-1)\delta x \leq |\lambda| < x\delta x \tag{5}$$

Avec $\delta x = 0.025$ et on normalise pour chaque boîte ainsi :

$$N_{norm}(x) = \frac{N(x)}{2\pi x \delta x} \tag{6}$$

On observe que pour les courbes normalisés et non normalisés, $N(x)$ décroît exponentiellement, pour les humains ce nombre décroît très fortement avec une pente de -0.65 quant à Gnugo plus faiblement avec -0.11. L'exposant n'est pas constant pour les humains.

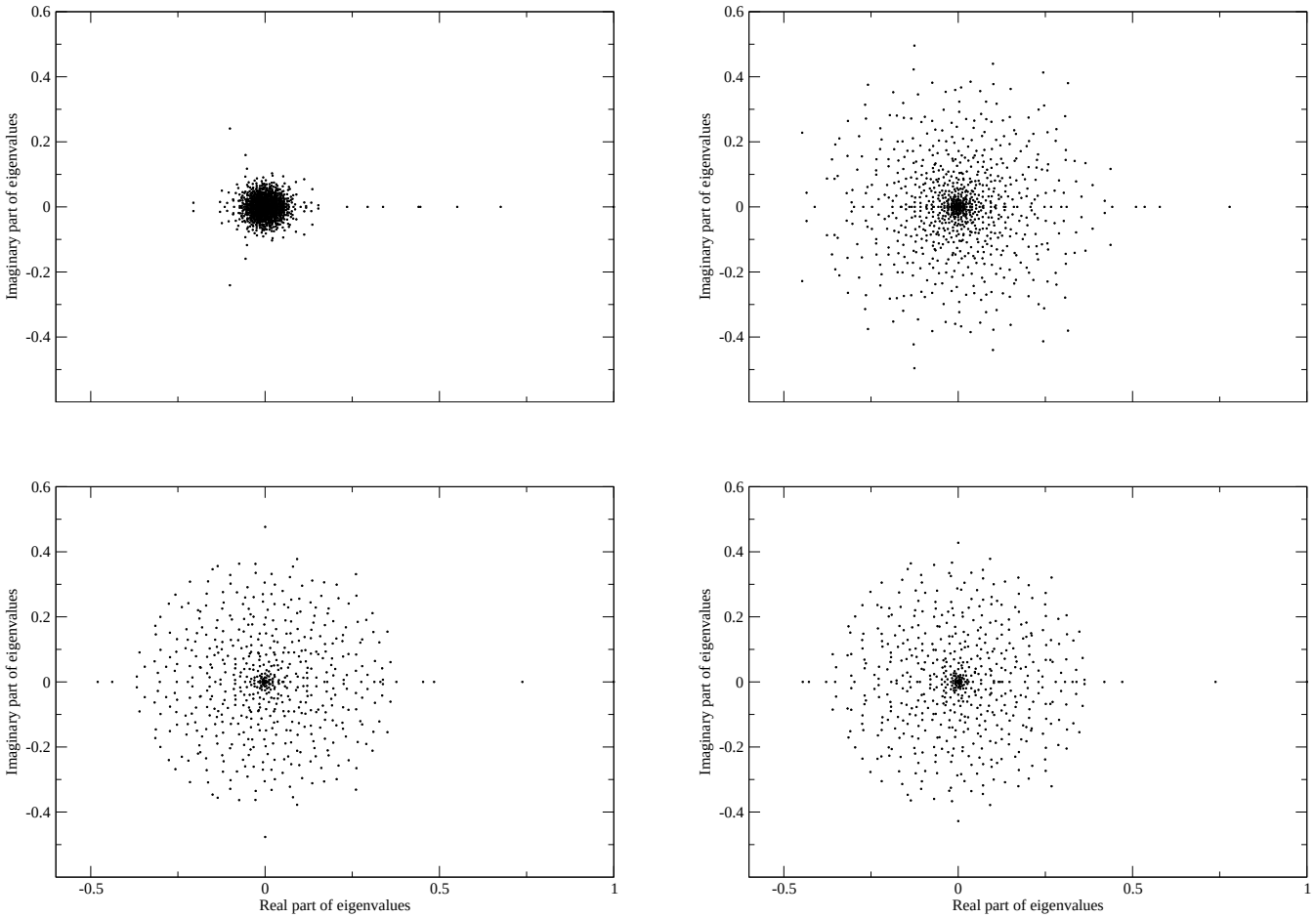
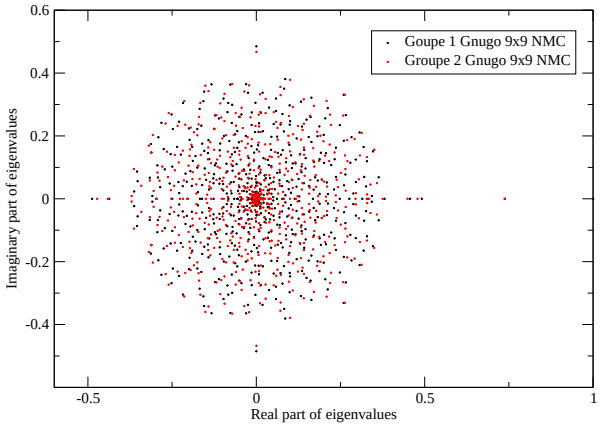
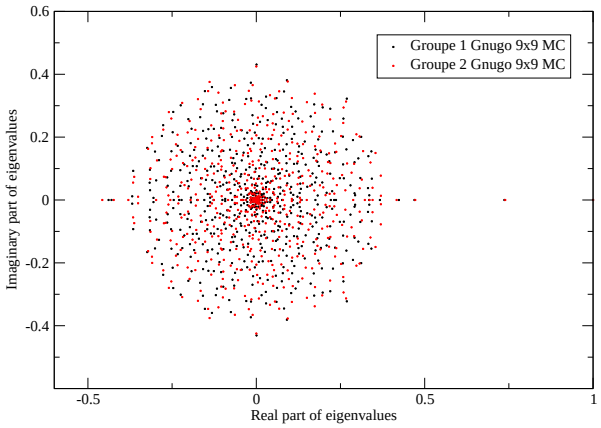


FIG. 5 – Spectres des valeurs propres dans le plan complexe pour G avec $\alpha = 1$, en haut à gauche U-go (humain) 19×19 4000 parties et en haut à droite Gnugo (ordinateur) 19×19 4000 parties, en bas à gauche Gnugo (ordinateur) 9×9 non Monte-Carlo 20 000 parties et en bas à droite Gnugo (ordinateur) 9×9 Monte-Carlo 20 000 parties. On voit une différence nette entre ces deux réseaux, la zone dense en valeurs propres est bien plus petite pour U-go (humain que pour tous les Gnugo (ordinateur)





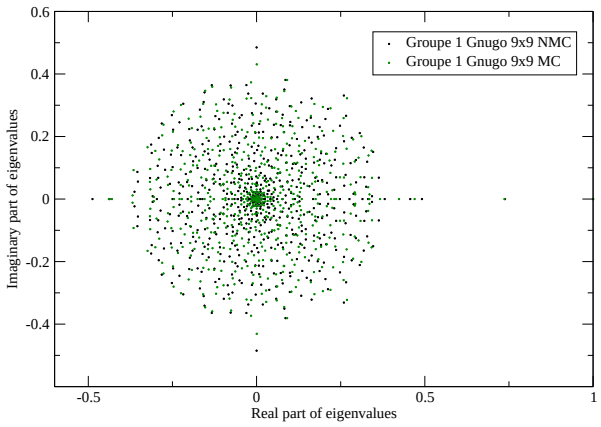


FIG. 6 – En haut spectres des valeurs propres dans le plan complexe pour G avec $\alpha = 1$ Gnugo (ordinateur) 9×9 non Monte-Carlo 2×10000 parties au milieu pour Gnugo (ordinateur) 9×9 Monte-Carlo 2×10000 parties, le nuage de point noir est relatif au groupe 1 et le nuage rouge au groupe 2 de chaque type d'algorithme. Les spectres sont étalées pour les quatre réseaux et on voit nettement que les spectres coïncident à en haut et au milieu, il n'y a donc pas de différence entre deux échantillons de même type d'algorithme. En bas il s'agit des spectres des valeurs propres dans le plan complexe de Gnugo (ordinateur) non Monte-Carlo contre Gnugo Monte-Carlo 9×9 de 10 000 parties pour G avec $\alpha = 1$, en noir le groupe 1 non Monte-Carlo en vert le groupe 1 Monte-Carlo. Il n'y a pas de différences visibles entre types Monte-Carlo et non Monte-Carlo.

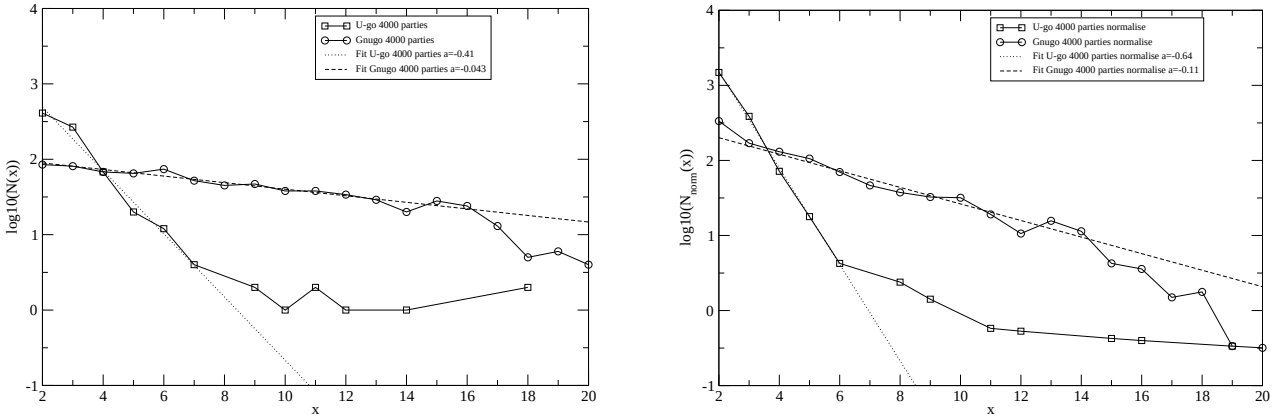


FIG. 7 – Histogrammes des valeurs propres dans la zone dense en valeurs propres en traçant le nombre des valeurs propres en logarithme décimale normalisé ($\log_{10}(N_{norm}(x))$) à droite et non normalisées à gauche ($\log_{10}(N(x))$) en fonction de x pour 4000 parties Gnugo (ordinateur) avec les cercles noirs et 4000 parties U-go (humain) avec les carrés noirs 19×19. La droite en pointillé noire est le fit pour U-go (humain) de pente -0.41 dans le cas non normalisé et -0.64 dans le cas normalisé et la droite en tiret noire est le fit pour Gnugo (ordinateur) de pente -0.043 dans le cas non normalisé et -0.11 dans le cas normalisé. Le spectre pour ordinateur a une zone dense uniforme tandis que pour les humains tout se concentre dans un cercle de rayon 0.05.

6 PageRank, CheiRank et autres Vecteurs Propres

Nos matrices de Google étant des matrices réelles et non symétriques, elles ont alors deux ensembles de vecteurs propres, les vecteurs gauches et les vecteurs droits, nous travaillerons avec les vecteurs droits qui sont des vecteurs colonnes. Les vecteurs propres droits nous permettent de classer les différents nœuds de nos réseaux, on observe une suite classée par importance décroissante des coups pour différentes communautés de coups et pour différentes phases de jeu[9]. Pour d'autres types de réseaux, comme celui construit à partir des pages wikipédia sur les universités mondiales, on peut grâce au PageRank classer les universités par leur importance[15]. Le vecteur propre droit correspondant à la plus grande valeur propre est appelé *PageRank*, il est le classement des coups les plus pertinent et les plus importants d'une banque de données, il ne s'agit pas nécessairement des coups les plus fréquemment joués. Le vecteur *CheiRank* est relié aux coups offrant le plus d'ouverture.

Les top 10 pour les différents types de joueurs pour le PageRank voir fig.8. Cette figure nous permet de constater "visuellement" la différence entre humain et ordinateur dans les différents choix stratégiques. La fig.9 nous montre que les coups les plus importantsà jouer dans un plus petit goban sont différent du grand goban. Cependant nous n'arrivons pas ici a différencier l'option Monte-Carlo du non Monte-Carlo. La même chose a été faite pour le CheiRank voir annexe.

Pour les autres vecteurs propres droits, en général ils représentent des familles de coups tendant à être joués ensemble et créant des structures dans le réseau (communauté)[9]. Nous voyons apparaître plus de plaquettes "bord" ou bien "coin" et aussi des plaquettes montrant des événements de type **"ko"** nom japonais pour **"éternité"**, c'est une configuration de pierres permettant la capture d'une pierre unique, et qui permettrait, si la règle du jeu s'y opposait pas la capture en retour immédiate d'une autre pierre, cet échange de captures ramenant à la position initiale[16] voir fig.19 en annexe. la fig.10 nous montre pour les humains amateurs plus de structures de réseau impliquant le ko contrairement à Gnugo avec plus de structures impliquant de longues chaînes.

Le rapport de participation inverse est utilisé principalement par les physiciens de la matière condensée afin de quantifier la localisation d'un vecteur propre et ça nous est utile pour voir les différences dans la structure des différents réseaux que nous avons obtenus. Ce rapport à déjà été utilisé pour l'étude de réseaux dirigés comme le web[17].

$$\mathrm{IPR} = \frac{(\sum_{i=1}^{N_v} |\psi_i|^2)^2}{\sum_{i=1}^{N_v} |\psi_i|^4} \tag{7}$$

avec

$$|\lambda| = e^{-2\gamma} \tag{8}$$

Soit ψ un vecteur propre d'une matrice M carrée de taille $N \times N$, prenons deux cas extrêmes :

- $\psi_i = 1 \ \forall i$ alors d'après (7) on a IPR $= N$ pour le cas d'une distribution d'occupation uniforme.
- $\psi_k = 0 \ \forall k \in [1; N-1]$ et $\psi_N = 1$ alors on a IPR $= 1$ pour le cas d'une distribution d'occupation non uniforme et singulière.

Nous avons tracés ce rapport de participation inverse pour tous les vecteurs propres pour nos différents réseaux en fonction de γ (8) qui nous permet de pouvoir mieux voir ce qui se passe pour les vecteurs propres correspondants aux valeurs propres proches de 0. On voit bien sur la fig.11 que les nuages de points ne se confondent pas, mais ce décalage est dût à l'éclatement du spectre des valeurs propres pour Gnugo, la valeur des rapports de participations inverses est globalement le même entre Gnugo (ordinateur) et U-go (humain). On peut distinguer les tailles 19×19 et 9×9 avec les fig.11 et fig.12 mais pas le type d'algorithme utilisé par la machine car même IPR, nous avons une preuve que les spectres de valeurs propres intra-algorithmes sont très proche car les nuages de points se confondent, voir fig.12.

Nous observons que pour tous les réseaux réciproques, le rapport de participation inverse est étiré vers les grandes valeurs de rapport on passe 80 à 120 pour le 19×19 voir fig.11 et de 60 à 80 pour le 9×9 voir fig.12. La différence dans les spectres de valeurs propres pour G et G^* n'étaient pas si différents, comparer avec les fig.21, 22, 23 et fig.24 en annexe.

Le spectre de valeurs propres est une donnée très motivante, nous permettant de bien distinguer Gnugo (ordinateur) de U-go (humain), cependant nous n'avons pas encore pu distinguer une différence entre les parties simulées grâce à un algorithme déterministe de celles simulées avec un algorithme Monte-Carlo. Le rapport de participation inverse est une donnée qui nous informe sur la quantification de la localisation des vecteurs propres associés aux différentes matrices de Google ne nous donne pas plus de différence entre les différents réseaux. Afin d'aboutir à un test de Turing pour le go nous utiliserons les vecteurs propres avec d'autres outils plus fins que le rapport de participation inverse.

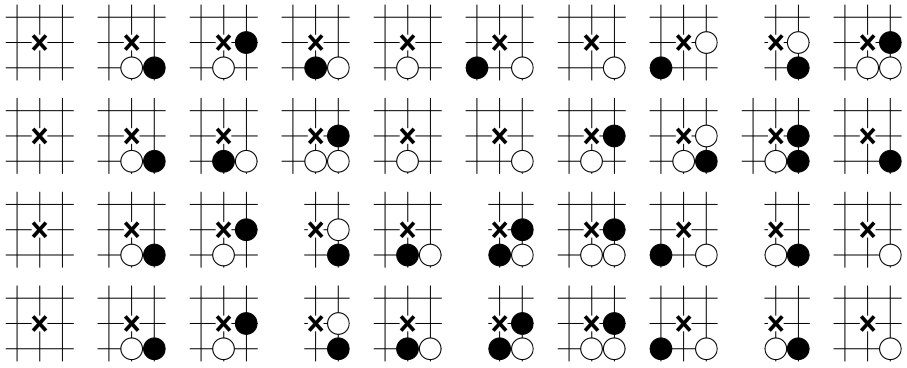


FIG. 8 – Top 10 des PageRank avec le joueur noir jouant sur la croix, de haut en bas Gnugo 19×19 (parties ordinateurs), parties amateurs 19×19 U-go(parties humaines), Gnugo 9×9 sans et avec option Monte-Carlo

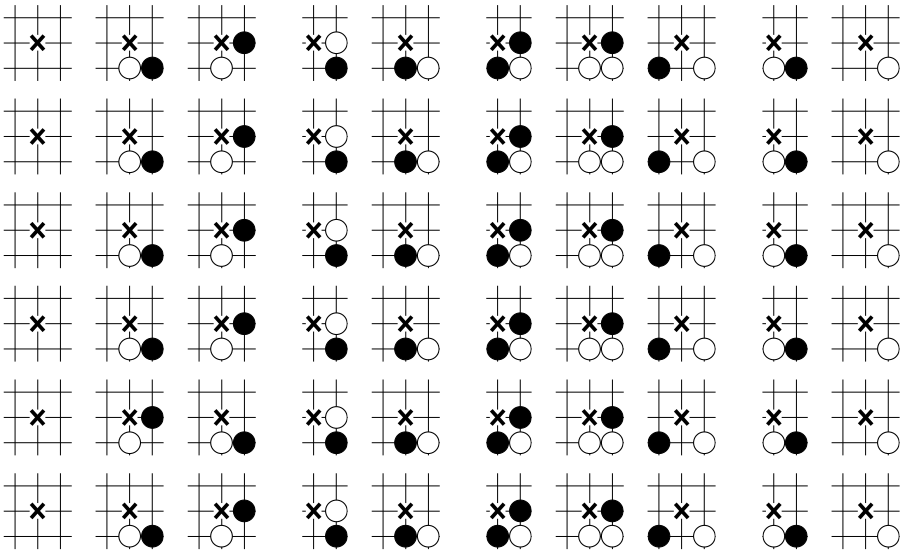


FIG. 9 – Top 10 des PageRank avec le joueur noir jouant sur la croix, de haut en bas 9×9 20 000 parties sans Monte-Carlo, avec Monte-Carlo, 9×9 Groupe 1 de 10 000 parties, Groupe 2 de 10 000 parties et 9×9 option Monte-Carlo Groupe 1 et groupe 2 de 10 000 parties chacun

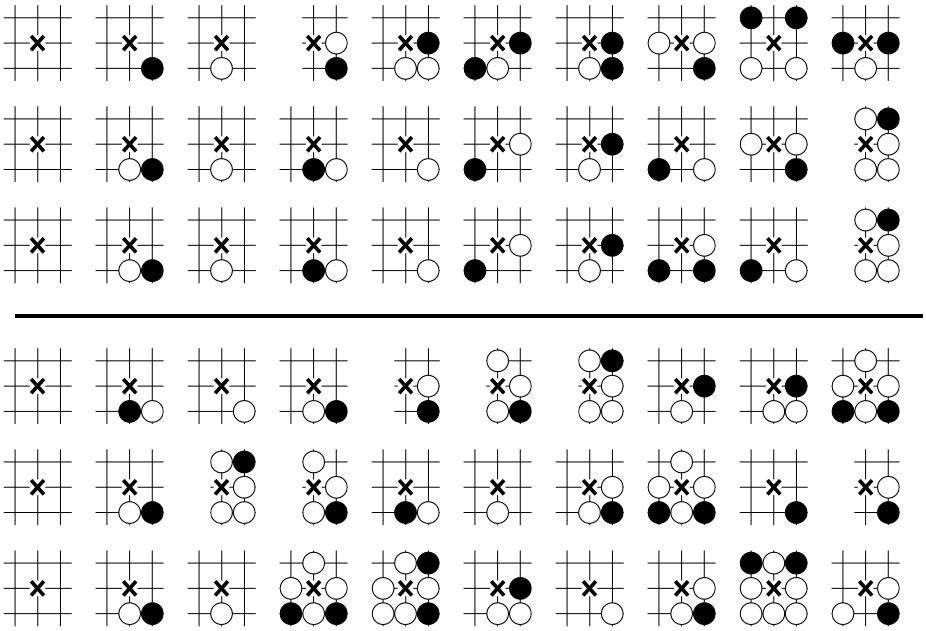


FIG. 10 – Les top 10 des plaquettes, avec le joueur noir jouant sur la croix, de 4000 parties Gnugo 19×19 en haut et Ugo 19×19 en bas, pour 3 autres vecteurs propres, de haut en bas leurs valeurs propres associées : λ_2, λ_3 et λ_4. On voit une différence stratégique dans la façon de créer des territoires, Gnugo (ordinateur) en haut lie des pierres entres elles tandis que U-go (humain) possède plus de plaquettes de type "ko".

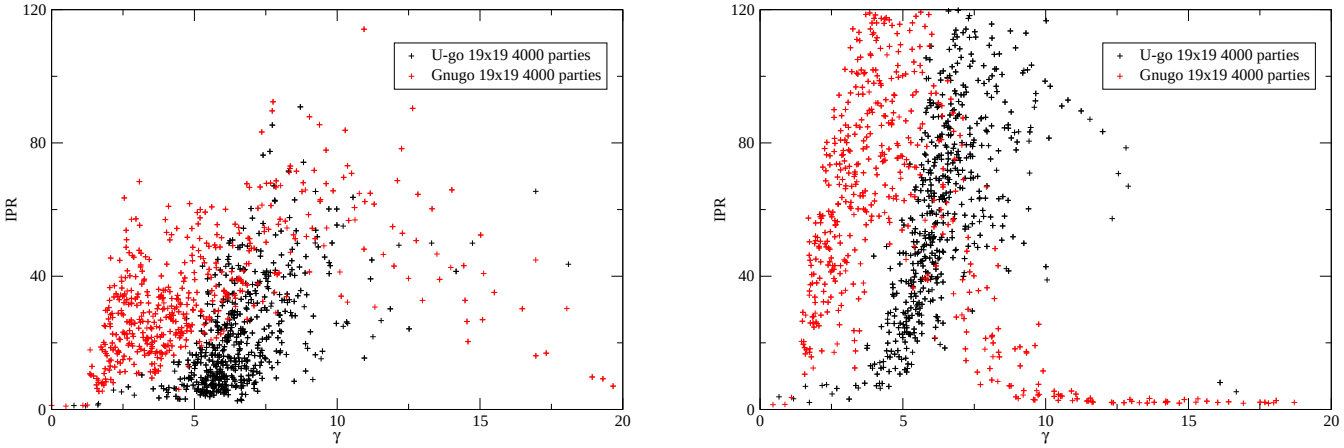


FIG. 11 – Pour Gnugo (ordinateur)et U-go (humain)19×19 de 4000 parties, il s'agit du rapport de participation inverse (IPR) pour chaque vecteur propre droit en fonction de γ qui dépend de la norme de la valeur propre associée, en rouge Gnugo et en noir U-go, à gauche pour G et à droite pour G^* avec $\alpha = 1$. On voit nettement une différence entre U-go, dont le nuage est décalé vers la droite et Gnugo. Pour la figure de droite on voit aussi un étirement vers les grandes valeurs de l'axe des abscisses pour les deux types de joueurs.

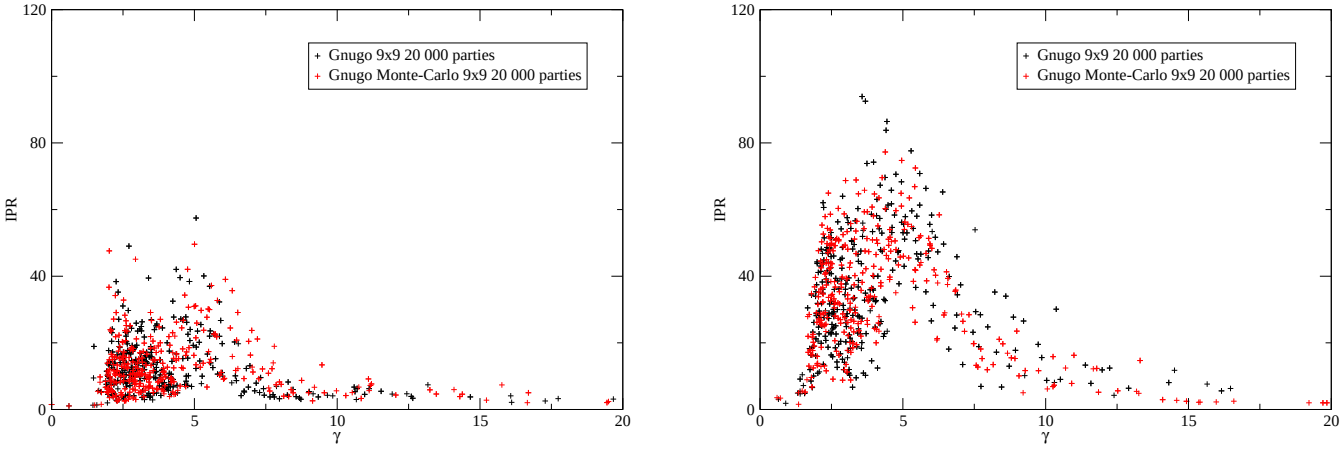


FIG. 12 – Pour Gnugo (ordinateur) 9×9 avec et sans option Monte-Carlo de 20 000 parties, il s'agit du rapport de participation inverse (IPR) pour chaque vecteur propre droit en fonction de γ qui dépend de la norme de la valeur propre associée, en rouge Monte-Carlo et en noir non Monte-Carlo, à gauche pour G et à droite pour G^* avec $\alpha = 1$. Les points rouges et les points noirs se confondent, il n' y a donc pas de différences entre types d'algorithme ici, et on a dans la figure de droite un étirement vers les grandes valeurs de l'axe des abscisses pour les deux types d'algorithmes.

7 Vers un test de Turing pour le go

Grâce aux résultats préliminaires vue précédemment, nous pouvons réfléchir à un algorithme nous permettant de quantifier la différence entre différents réseaux de parties de go. Les spectres de valeurs propres montre clairement une différence entre les réseaux voir fig.5.

Trois tests de Turing pour le Go :

– La fidélité, grandeur associée au produit scalaire entre le PageRank pour un type de joueur et l'étalon Gnugo (ordinateur) ou bien l'étalon U-go (humain).

$$\text{Fidélité} = \sum_{i=1}^{N_v} \alpha_i \beta_i \tag{9}$$

où α_i et β_i sont respectivement les ièmes composantes des ψ_1 pour un sous groupe et le groupe étalon. N_v est le nombre de nœuds du réseau.

– La similarité de PageRank, où ici on calcul le pourcentage de ressemblance entre les 30 premières entrées du PageRank pour un type de joueur et celui de l'étalon Gnugo (ordinateur) ou bien U-go (humain).

$$\text{Similarité de PageRank} = \sum_{i=1}^{30} \frac{f(i)}{30} \tag{10}$$

$$f(i) = \begin{cases} 0 & \text{si } A_i \neq B_i \\ 1 & \text{sinon.} \end{cases} \tag{11}$$

où A_i et B_i sont respectivement les ièmes composantes des PageRank pour un sous groupe et le groupe étalon. Cette quantité nous dis si une des 30 meilleurs entrées est différentes entre le classement étalon et d'un sous échantillon.

– Le coefficient de corrélation entre PageRank, il s'agit ici de tracer la corrélation entre le PageRank d'un type de donnée et celui d'une autre type de joueur et de calculer le coefficient de corrélation, qui est ni plus ni moins ici que la distance moyenne à la diagonale.

$$\sigma = \sqrt{\sum_{i=1}^{N_v} \frac{(PrA_i - PrB_i)^2}{N_v}} \tag{12}$$

où PrA_i est la composante i du PageRank du groupe A et respectivement PrB_i est la composante i pour le groupe B.

7.1 Fidélité et similarité et spectre

Nous pensons qu'en prenant un nombre de partie assez conséquent pour chaque type de joueur, nous aurons une statistique tel que dans l'ensemble des parties ordinateurs ou humaines, chacun des sous groupes auraient un PageRank proche de l'étalon correspondant. Le PageRank étalon se calcul sur le grand ensemble. Pour les amateurs 8000 parties, pour Gnugo 19×19 8000 parties et pour Gnugo 9×9 non Monte-Carlo et Monte-Carlo 20 000 parties.

Ce test marche très bien pour distinguer U-go de Gnugo pour tous nos groupes voir fig.13, on peut aussi faire la moyenne des fidélités et similarités pour différents vecteurs propres droits correspondant aux 7 premières valeurs propres voir fig.29 en annexe.

Pour ce qui est du test appliqué au 9×9 non Monte-Carlo et Monte-Carlo, il est négatif comme on peut le voir sur la fig.14 qui ne permet pas de distinguer les deux types d'algorithmes.

Pour le cas du spectre, on peut approfondir son analyse en observant l'évolution de la zone regroupant 80 % des valeurs propres et celle regroupant 90 % des valeurs propres pour différents nombres et types de parties et pour différentes conditions sur λ_c. Avec $\lambda_{c,80}$ et $\lambda_{c,90}$ tel que : $\#\lambda \leq \lambda_{c,80} = 0.8 \times \#_{\text{total}}\lambda$ et de même pour $\lambda_{c,90}$.

On observe alors que ce rayon est sensible au nombres de parties pour le cas des humains, mais pour l'ordinateur il y a une certaine constance fig.15.

7.2 Coefficient de corrélation

Ici pour différentes tailles d'échantillons correspondant à différents types de joueurs et d'algorithmes nous voulons voir si la largeur de la courbe de corrélation entre PageRank change d'un réseau à l'autre, si c'est le cas on pourrait utiliser le coefficient de corrélation pour différencier les types de parties que nous avons.

On observe une grande différence entre corrélations de PageRanks entre Gnugo (ordinateur) et U-go (humain) tandis que pour le petit goban, la différence entre corrélations de PageRanks est marqué par le changement de la taille d'échantillon, voir fig.16. Il n'y a pas de différence entre Gnugo non Monte-Carlo et Gnugo Monte-Carlo voir fig.30 en annexe.

On ne peut pas dire la même chose en ce qui concerne les types d'algorithmes fig. 30. Le "temps" moyen d'utilisation des plaquettes les moins bien classé du PageRank est tellement petit et proche que pour éviter que l'algorithme de classement ne biaise pas l'ordre, on prend la moitié soit 553 au lieu des 1107 entrées pour les différents PageRank a corréler.

7.3 Fidélité et coefficient de corrélation

Les différents groupes de parties sont les suivants :

19×19 : Gnugo avec un étalon pour 8000 parties, deux groupes de 4000 parties et 8 sous groupes de 1000 parties et U-go (humain) avec un étalon pour 8000 parties, deux groupes de 4000 parties et 8 sous groupes de 1000 parties.

9×9 : non Monte-Carlo avec un étalon de 20 000 parties, deux groupes de 4000 parties et 8 sous groupes de 1000 parties et Monte-Carlo avec un étalon de 20 000 parties, deux groupes de 4000 parties et 8 sous groupes de 1000 parties.

On voit très bien que la fidélité couplée au coefficient de corrélation voir fig.17 nous permet de distinguer le groupe des humains et des ordinateurs et nous permet de distinguer les tailles d'échantillons pour le cas Gnugo 9×9, il semblerait pour ce dernier cas, que 20 000 parties ne soit pas un bon étalon pour le 9×9, ou bien que le simulateur a un algorithme de type Monte-Carlo peu performant et présente sur une taille 9×9 de goban aucune différence avec son algorithme par défaut qui lui est déterministe.

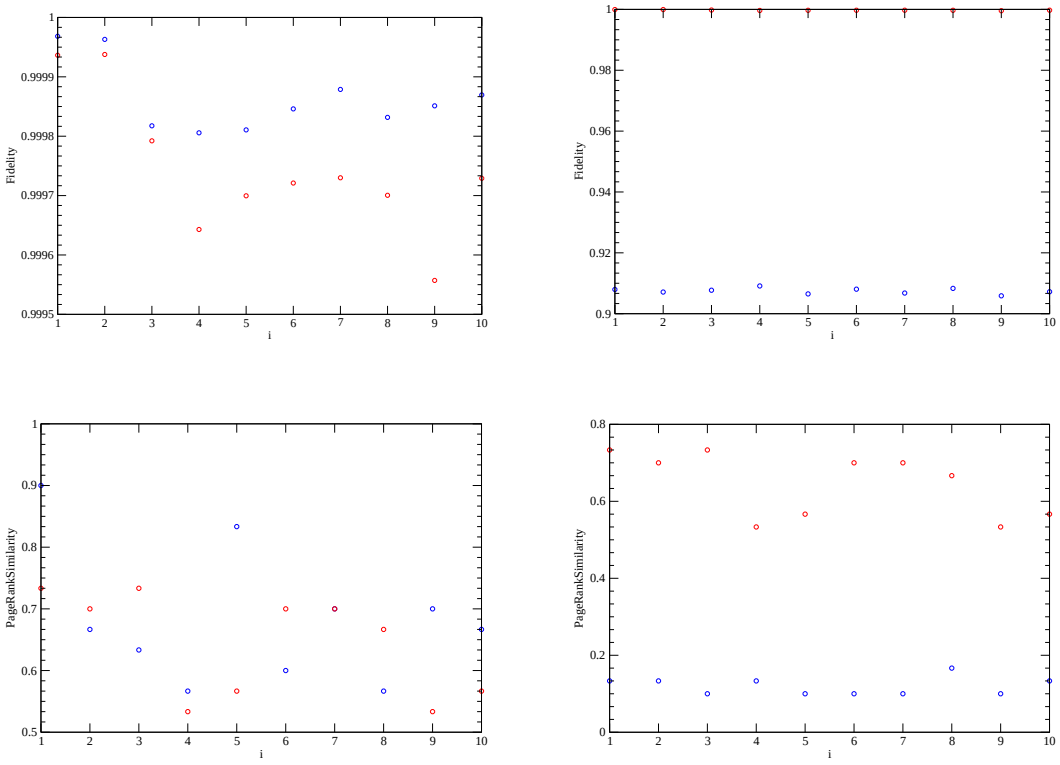


FIG. 13 – Tests de Turing avec Gnugo (ordinateur) et U-go (humain) 19×19 : les figures du haut sont pour la fidélité pour le vecteur propre droit ψ_1 pour différentes banques de données, pour i = 1 et 2 nous avons 4000 parties , i>2 nous avons 1000 parties, à gauche en rouge U-go avec étalon U-go et en bleu Gnugo avec étalon Gnugo et à droite en rouge U-go avec étalon U-go et en bleu Gnugo avec étalon U-go. Les figures du bas sont pour la similarité de PageRank. On voit nettement que humain et ordinateur se distinguent facilement avec ces deux tests, les figures de gauche ne permettent pas de distinguer rouge et noir mais en utilisant un même étalon (étalon humain) on peut voir que il y a à droite une séparation nette entre rouge et bleu.

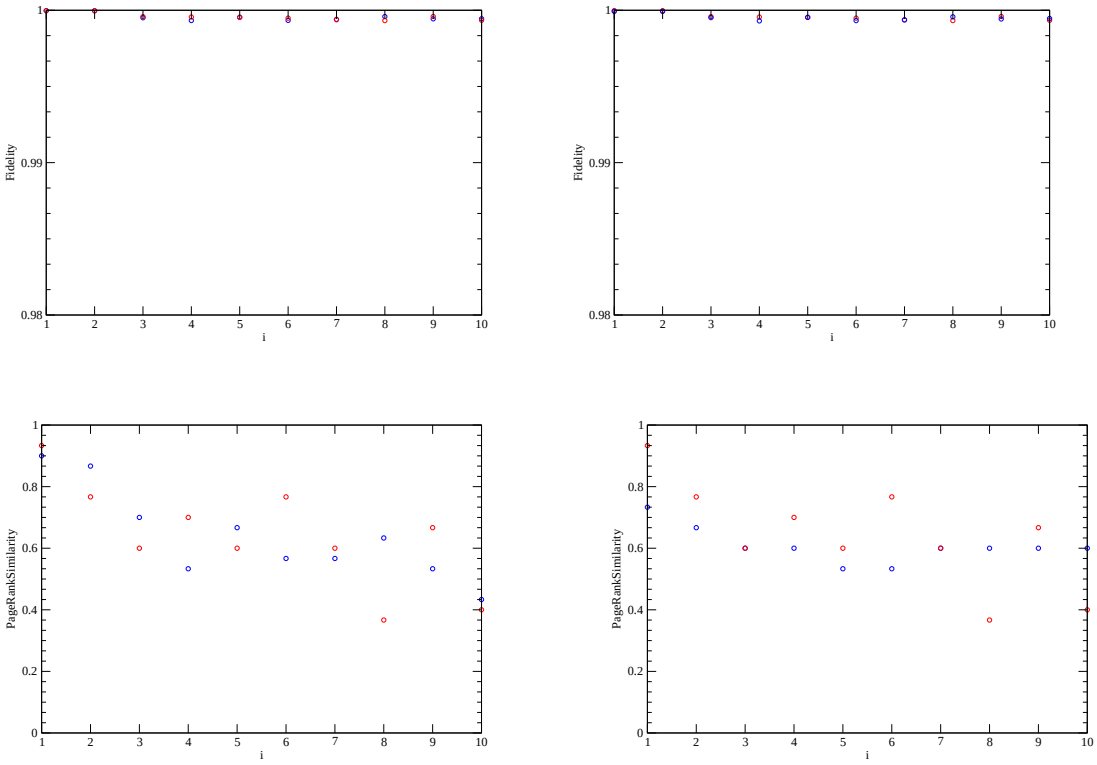


FIG. 14 – Tests de Turing avec Gnugo (ordinateur) Monte-Carlo et non Monte-Carlo 9×9 : les figures du haut sont pour la fidélité pour le vecteur propre droit ψ_1 pour différentes banques de données, pour i = 1 et 2 nous avons 10 000 parties , i>2 nous avons 1000 parties, à gauche en rouge Non Monte-Carlo avec étalon non Monte-Carlo en bleu Monte-Carlo avec étalon Monte-Carlo et à droite en rouge non Monte-Carlo avec étalon non Monte-Carlo en bleu Monte-Carlo avec étalon non Monte-Carlo. Les figures du bas sont pour la similarité de PageRank. On voit clairement que pour la fidelité, les figures gauche et droite ne montre aucune différences et tous les groupes de données semblent être tous suivre l'étalon Monte-Carlo et non Monte-Carlo. Pour les figures du bas aucune différences entre gauche et droite et les similarités varient d'un groupe à l'autre, à droite rouge et bleu se confondent pour i=3 et i=7.

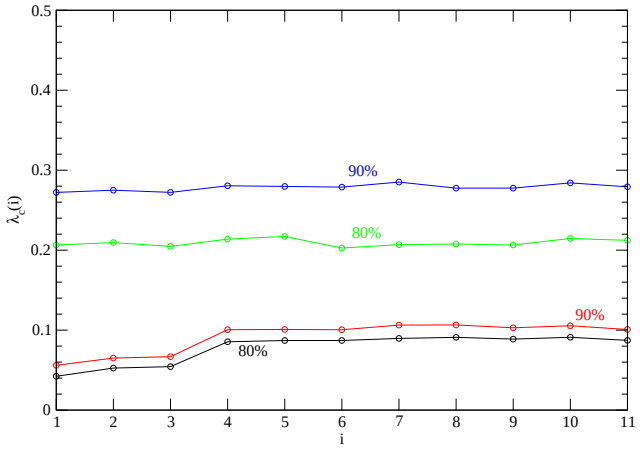


FIG. 15 – Variation de λ_c, rayon limite contenant 80% des valeurs propres(courbes en bas et 90% courbe du haut), pour différents groupes de parties U-go/Gnugo 19×19. i = 1 avec 8000 parties, i=2 et i=3 avec 4000 parties et de i=4 à i=11 avec 1000 parties en bleu (90%) et vert (80%) Gnugo (ordinateur) et en rouge (90%) et noir (80%) U-go (humain). On voit très clairement que pour Gnugo il y a un stabilité pour différentes tailles de banque de données avec $\lambda c \approx 0.21$ pour la courbe verte et $\lambda c \approx 0.27$ tandis que pour U-go il y a plusieurs plateau de $\lambda c \approx 0.06$ à $\lambda c \approx 0.1$ pour la rouge et de $\lambda c \approx 0.04$ à $\lambda c \approx 0.8$ pour la noire.

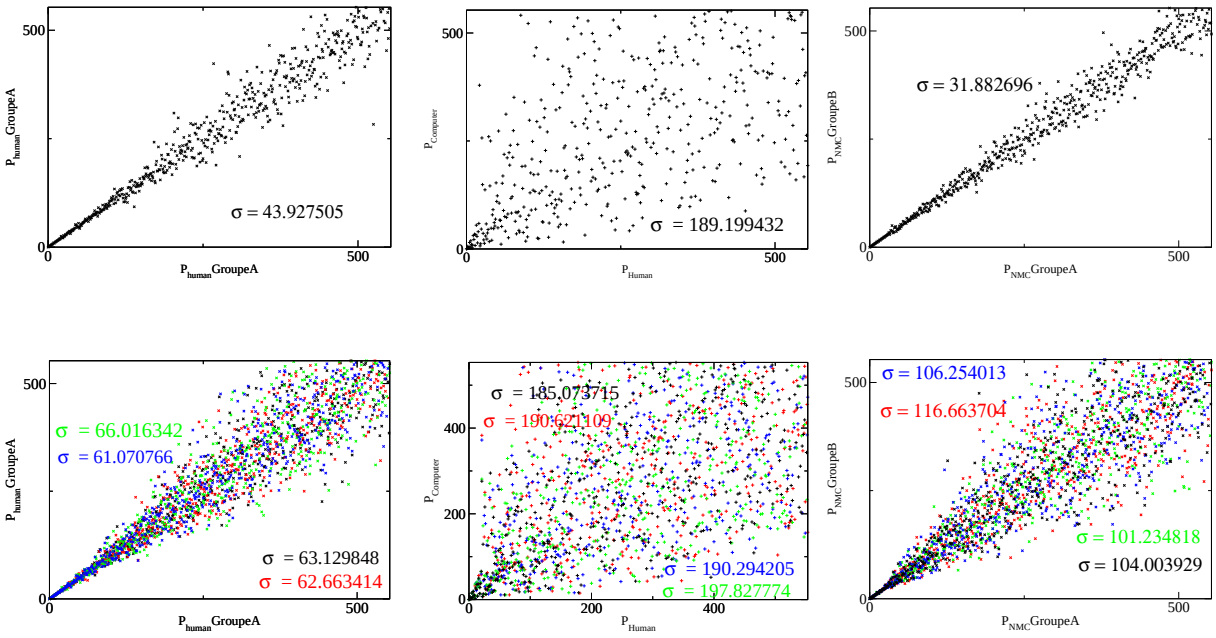


FIG. 16 – En haut à gauche corrélation entre PageRanks U-go (humain) 9×9 contre U-go (humain) 9×9 pour deux groupes de 4000 parties au milieu U-go 9×9 (humain) contre Gnugo (ordinateur) 9×9 avec deux groupes de 4000 parties et à droite Gnugo 9×9 non Monte-Carlo contre non Monte-Carlo pour des groupes de 10 000 parties. En bas mêmes choses mais pour 4 couples de 1000 parties donc à gauche U-go contre U-go au milieu U-go contre Gnugo et à droite non Monte-Carlo contre non Monte-Carlo. Ces corrélations de PageRanks évoluent différemment pour les parties Gnugo et U-go, les nuages de points fuient la diagonale au passage des figures homotypes aux figures hétérotypes, tandis que dans le cas de Gnugo 9×9 le changement de la taille d'échantillon semble être en être responsable.

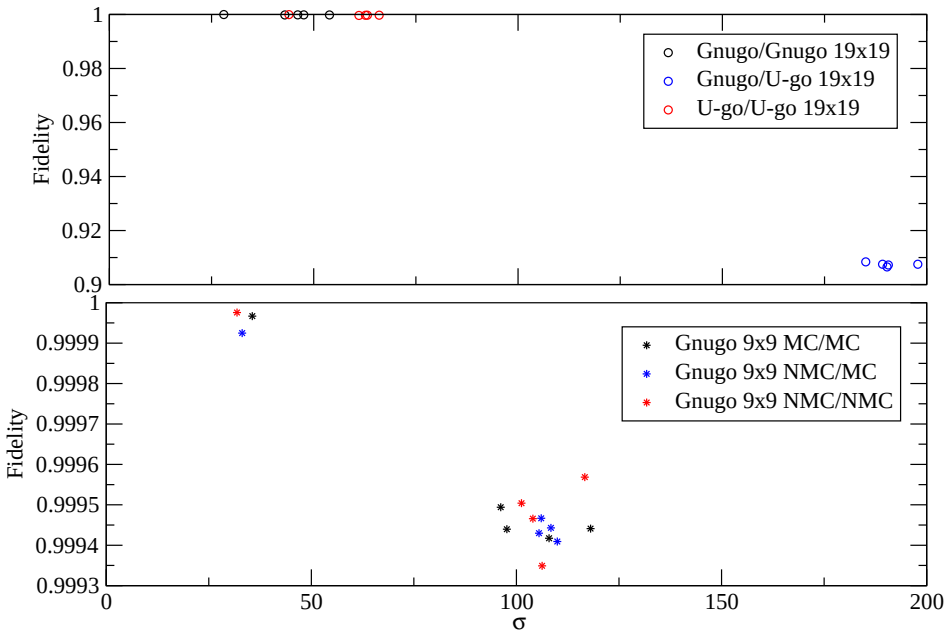


FIG. 17 – Un dernier test de Turing couplant fidélité et coefficient de corrélation pour les différents réseaux 19×19 en haut et 9×9 en bas. On a différents types de points, les cercles pour le grand goban et les étoiles pour le petit, en haut on voit que le test croisé en bleu se dissocie du reste des points et donc on distingue le type de joueur, en revanche en bas le test croisé en bleu reste avec les autres tests en rouge et noir mais nous avons deux groupes de points qui est due aux tailles des échantillons.

Cinquième partie

Discussion

Les réseaux que nous avons obtenus à partir des banques de données U-go (humain) et Gnugo 19×19 et 9×9 (ordinateur), sont des réseaux denses et avec une distribution intégrée de liens qui est symétrique entre les liens entrants et sortants. Les pentes de ces distributions sont caractéristiques des réseaux invariants d'échelles et sont souvent qualifié de *"petits mondes"*. Un petit monde est un réseau avec $\bar{l}_{path}$ la longueur de plus court chemin moyenne entre V_i et V_j deux nœuds du réseau est petite.

La différence pour les distributions intégrées entre la structure d'un réseau de parties humaines de celle d'un réseau de parties Gnugo est inexistante, les deux réseaux suivent une même loi de puissance. En revanche pour le petit goban les pentes sont bien différentes, on passe de ≈ -1 pour le 19×19 à ≈ -0.8 pour le 9×9. Nous n'avons pu mettre en évidence qu'une faible différence entre les distributions intégrées des liens pour deux réseaux de go 9×9 de types d'algorithmes différents les pentes sont très proche -0.84 pour le type déterministe et -0.80 pour le type Monte-Carlo. De plus les pentes fluctuent autour de ≈ -0.74 et ≈ -0.78 pour les liens entrants et sortants du type Monte-Carlo quand on réduit la banque de données à 10 000 au lieu de 20 000 parties. De même les pentes fluctuent autour de ≈ -0.82 et ≈ -0.80 pour les liens entrants et sortants du type déterministe. Le décrochage visible pour les hauts degrés des distributions sont présent uniquement dans le cas des réseaux Gnugo, ce qui est une première différence entre le groupe des réseaux Gnugo et celui du réseau humain. Il s'agit peut être là d'un effet due à la façon de jouer du programme, une sorte "d'habitude". Une explication pour la différence de pente entre format de goban pourrait être que dans un plateau plus petit, les coups sont joués plus proche et donc on a vite plus de liens.

Les spectres de valeurs propres nous apprend plus sur la différence entre humain et Gnugo qu'entre type d'algorithme utilisé dans la simulation. La différence majeur est la zone dense en valeurs propres, en effet pour les humains cette zone est plus concentrée que pour Gnugo, il y a 8 valeurs propres réelles et positives pour les humains tandis que pour Gnugo il y a plus de valeurs propres réelles et positives. La

différence entre humain et Gnugo pourrait être que pour 8000 parties humaines nous avons au mieux 16 000 joueurs différents tandis que pour Gnugo un seul simulateur jouant contre lui-même 8000 fois.

Les PageRanks associés aux matrices Google de chaque réseau sont identifiable comme étant la liste des meilleurs coups présents dans une banque de données. Pour les autres vecteurs propres on y voit d'autres familles de coups, par exemple les phases de ko, des motifs relatifs à l'élaboration de chaînes de pierres ou bien encore des plaquettes jouées aux bords du goban. Il semblerait que Gnugo joue plus de chaînes de pierres et dans les coins et les humains vont plus chercher le ko et reste dans le goban.

En changeant la taille des banques de données utilisées pour chaque réseau, la zone dense du spectre Gnugo (ordinateur) 19×19 reste identique mais pour le cas de U-go la zone dense du spectre s'élargit, les résultats sur les rayons pour 80 % et 90 % des valeurs propres données par λ_c nous le confirme.

Les tests de Turing pour le go, que nous avons élaborés, nous permettent de distinguer parfaitement Gnugo (ordinateur) et U-go (humain). Il semble que Gnugo n'imite pas la façon de jouer d'un humain. En revanche les différents algorithme utilisé par Gnugo semblent être similaires.

On peut différencier humain d'ordinateur pour le goban 19×19, cependant pour le petit goban la seule différence qu'on ait avec les trois tests de Turing est due aux différentes tailles de banque de données.

Sixième partie

Conclusion

Nous avons vu que la théorie des réseaux complexes, la physique statistique et la physique de la matière condensée, apportent plusieurs outils nous permettant d'étudier plusieurs type de système, que ce soit en biologie, en informatique et même en sociologie. Que ce soit par la forme du graphe construit, par la distribution intégrée des différent types de liens, par le spectre des valeurs propres, les valeurs propres et vecteurs propres associés à la matrices de Google nous permet de distinguer par exemple deux types d'individus via leurs réseaux de séquences ADN ou encore dans notre cas deux types de joueurs de go.

Il y a plus de différence entre humain et ordinateur qu'entre groupes d'humains. Cependant tout comme il y a peu de différence dans les spectres entre groupes d'humains, il y a peu de différence entre algorithme déterministe et Monte-Carlo.

Nous avons pu mettre en place différentes méthodes pour quantifier la différence entre humain et ordinateur dans ce jeu, on pourrait étendre cette études en changeant de type de réseau avec un réseau construit à partir de type de plaquette plus grand, on aura alors plus de nœud dans le réseau et on pourrait alors voir une différence entre Monte-Carlo et non Monte-Carlo.

Il serait aussi intéressant d'étendre ce test de Turing pour le go avec d'autres simulateurs.

Enfin cette méthode des réseaux complexes a encore beaucoup d'applications possibles, comme par exemple l'étude d'optimisation d'ARN, qui est bien étudié déjà mais pourrait être vue d'une toute autre façon sous la forme d'un réseau. Il reste encore à voir pour un réseau construit avec une autre norme de plaquette offrant un nombre de nœud plus grand et peut être nous pourrions encore plus affiner nos résultats.

Références

[1] Bertrand Georgeot and Olivier Giraud "The game of go as a complex network", Europhysics Letters 97, 68002 (2012).

[2] http ://worldwidewebsize.com

[3] Langville and Meyer, "Google's PageRank an Beyond THE SCIENCE OF SEARCH ENGINE RANKINGS", Princeton University Press 2006.

[4] V.Kandiah and D.L.Shepelyansky, "Google matrix analysis of DNA sequences", PLOS One v.8(5), p. e61519 (2013).

[5] G.K.Zipf, "The Psycho-Biology of language" (Houghton Mifflin, Boston) 1935.

[6] X.Gabais, Q.J. Econ., 114 (1999) 739.

[7] B.Blasius and R. Töner, Phys. Rev. Lett., 103 (2009).

[8] "Computing machinery and intelligence", Oxford University Press, vol. 59, 236 (1950).

[9] Vivek Kandiah, Bertrand Georgeot and Olivier Giraud, "Move ordering and communities in complex networks describing the game of go", Eur. Phys. J. B 87, 246 (2014).

[10] www.gnu.org/software/gnugo

[11] www.gnu.org/software/gnugo/gnugo_3.html

[12] www.u-go.net

[13] S.N.Dorogovtsev et J.F.F. Mendes, "Evolution of Networks From Biological Nets to the Internet and WWW " Oxford University Press 2003.

[14] D.Donato, L.Laura, S.Leonardi et S.Milozzi, Eur. Phys. J. B, 38 (20004) 239.

[15] J.Lages, A.Patt and D.L.Shepelyansky, "Wikipedia ranking of world universities", Eur. Phys. J. B v.89, p.69 (2016).

[16] https ://fr.wikipedia.org/wiki/Ko_(go)

[17] B. Georgeot, O. Giraud, and D.L. Shepelyansky "Spectral properties of the Google matrix of the World Wide Web and other directed network" Phys. Rev. E 81, 056109 (2010)

Septième partie

Annexe

8 Quelques règles du go

Le goban est le nom du plateau de jeu de go, il y a plusieurs tailles possibles et sont tous carrés. Il y a un nombre N de lignes horizontales et verticales. On place une pierre soit noire soit blanche sur une des intersections du goban, voici une figure représentant le goban pour une taille de 19×19 : Chacun

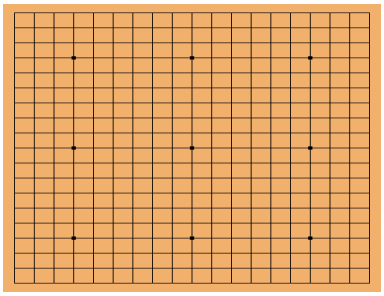


FIG. 18 – Un goban de taille 19×19 lignes, 19 horizontales et 19 verticales, il s'agit du plateau le plus utilisé en tournoi.

son tour les deux joueurs placent une pierre de couleur, le but du jeu est de créer de grand territoire afin de protéger ses pierres et d'en capturer. L' attari est un moment du jeu où un groupe de pierre ou une pierre d'un joueur est entouré par des pierres de l'autre joueur ne laissant qu'une seule liberté, voici une figure représentant cette phase de jeu.

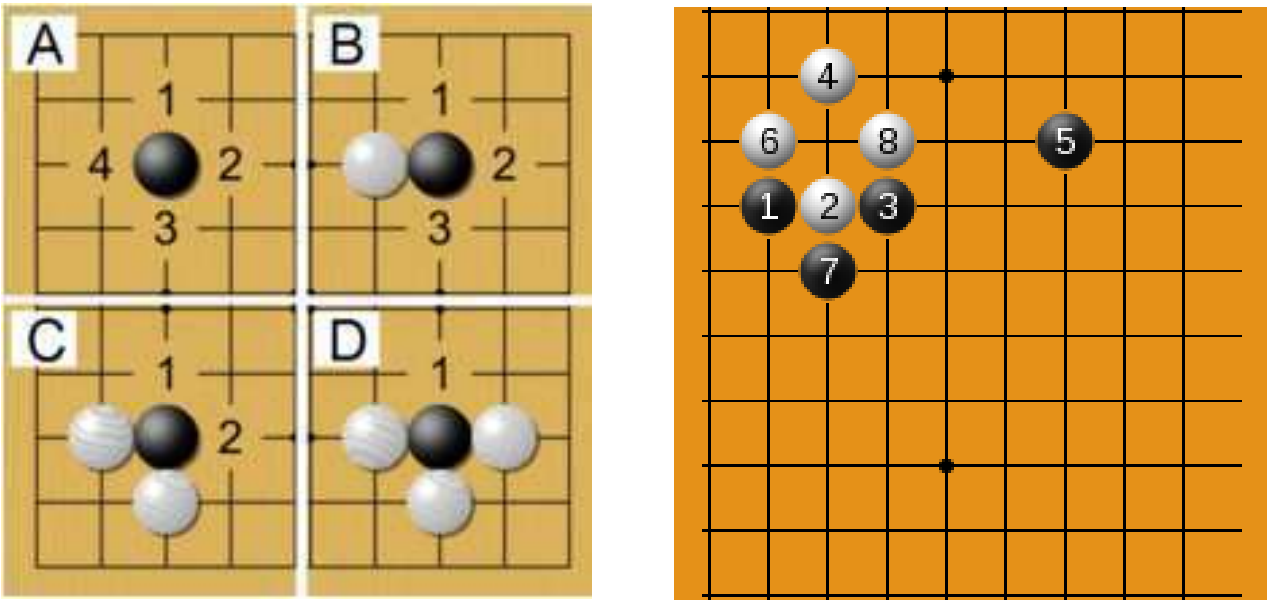


FIG. 19 – À gauche un exemple simple d'attari, les images A, B, C et D représente trois tours au bout desquels la pierre centrale est placé en attari. À droite un exemple simple de ko.

9 Archives de jeu

Voici un exemple de fichier Smart Game Format :

```
(;GM[1]
FF[4]
CA[UTF-8]
AP[CGoban:3]
ST[2]
RU[Chinese]
SZ[19]
KM[7.50]
TM[7200
]OT[3x60 byo-yomi]
PW[AlphaGo]
PB[Lee Sedol]
BR[9d]
DT[2016-03-15]
EV[Google DeepMind Challenge Match]
RO[Game 5]
PC[Seoul, Korea]
WT[Computer]
BT[Human]
SO[https://gogameguru.com/]
RE[W+Resign]
;B[qd];W[dd];B[pq];W[dp];B[oc];W[po];B[qo];W[qn];B[qp];W[pm];B[nq]
;W[qe];B[pe];W[qf];B[rd];W[pf];B[ql];W[oe];B[pl];W[ol];B[om];W[ok]
;B[nm];W[qj];B[rn];W[gq];B[cf];W[fc];B[bd];W[ch];B[dh];W[di];B[dg]
;W[cc];B[ci];W[cj];B[bi];W[dj];B[bh];W[ml];B[mm];W[lm];B[nl];W[nk]
;B[ln];W[ll];B[kn];W[mn];B[mo];W[rm];B[rl];W[ro];B[sn];W[pn];B[oo]
;W[op];B[no];W[sm];B[pp];W[nc];B[nb];W[ob];B[pb];W[od];B[pc];W[mb]
;B[oa];W[mc];B[gd];W[gf];B[gc];W[fd];B[ge];W[ff];B[hf];W[hg];B[if]
;W[ig];B[id];W[jf];B[jb];W[jd];B[jc];W[je];B[gh];W[fb];B[gg];W[fe]
;B[hc];W[hi];B[jl];W[hb];B[gb];W[ga];B[ib];W[ha];B[ia];W[fa];B[ka]
;W[iq];B[ii];W[hh];B[hj];W[gi];B[gj];W[cn];B[dq];W[cq];B[fq];W[eq]
;B[fp];W[gp];B[do];W[co];B[ep];W[dr];B[er];W[dq];B[gr];W[fo];B[eo]
;W[fn];B[hr];W[in];B[ir];W[jq];B[kr];W[jr];B[js];W[kq];B[lr];W[lq]
;B[mq];W[mr];B[nr];W[ik];B[fi];W[fh];B[jk];W[ij];B[jj];W[il];B[jm]
;W[im];B[eh];W[fg];B[mj];W[lj];B[li];W[mi];B[lk];W[nj];B[kj];W[ei]
;B[bk];W[bj];B[aj];W[is];B[hs];W[so];B[rn];W[rk];B[sl];W[ai];B[ah]
;W[cl];B[bl];W[bm];B[dm];W[dn];B[cm];W[dl];B[em];W[en];B[fl];W[fj]
;B[bn];W[ak];B[al];W[fr];B[mh];W[ni];B[lg];W[ks];B[ls];W[bc];B[bo]
;W[bp];B[og];W[mf];B[nh];W[ph];B[ce];W[mk];B[sg];W[rg];B[gl];W[re]
;B[se];W[sf];B[rf];W[sh];B[ld];W[me];B[ac];W[ab];B[ad];W[cd];B[bf]
;W[kc];B[kb];W[oh];B[nf];W[lc];B[bb];W[cb];B[aa];W[gk];B[hl];W[hk]
;B[jh];W[kg];B[lf];W[kh];B[lh];W[le];B[kf];W[jg];B[ao];W[lj];B[ki]
;W[jn];B[km];W[sn];B[on];W[rp];B[rq];W[pk];B[qm];W[sf];B[sd];W[ma]
;B[na];W[hq];B[ms];W[ko];B[lp];W[kp];B[lo];W[fs];B[ks];W[el];B[fm]
;W[hm];B[am];W[kk];B[ne];W[of];B[de];W[ie];B[he];W[ap];B[ee];W[ed]
;B[ke];W[kd];B[pd];W[pg];B[ng];W[sk];B[rn];W[fk];B[kl];W[gm];B[bm]
;W[lk];B[ck];W[dk];B[qk];W[rj])
```

FIG. 20 – Un exemple de fichier sgf du dernier matche entre le 3$^{\text{ème}}$joueur mondial Lee Sedol et le simulateur AlphaGo de DeepMind. WR et BR indiquent le classement des joueurs blanc et noir, HA indique la présence de pierre de handicap et leur AB est leur emplacement. Les B/W suivit du couple de lettre entre crochet est la position de la pierre joué par le joueur noir/blanc. SZ est la taille du goban. RE donne une information sur qui a gagné et comment il a gagné, ici "par abandon".

10 Figures

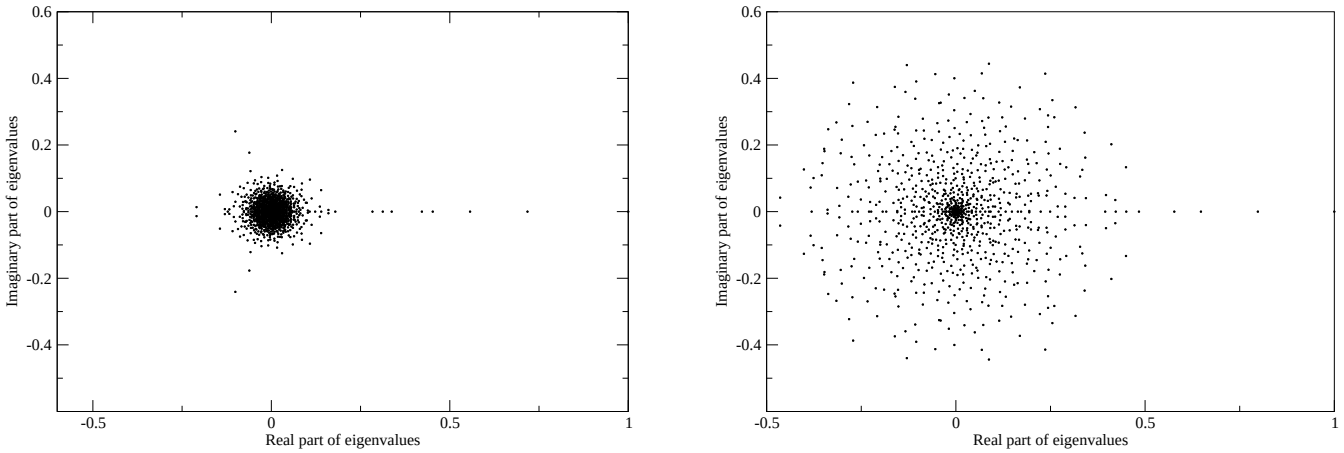


FIG. 21 – Spectres des valeurs propres dans le plan complexe pour G^* avec $\alpha = 1$, à gauche U-go (humain) 19×19 4000 parties et à droite Gnugo (ordinateur) 19×19 4000 parties, on voit une différence nette entre ces deux réseaux, la zone dense en valeurs propres est bien plus petite pour U-go que pour Gnugo. Il y a plus de structures pour Gnugo.

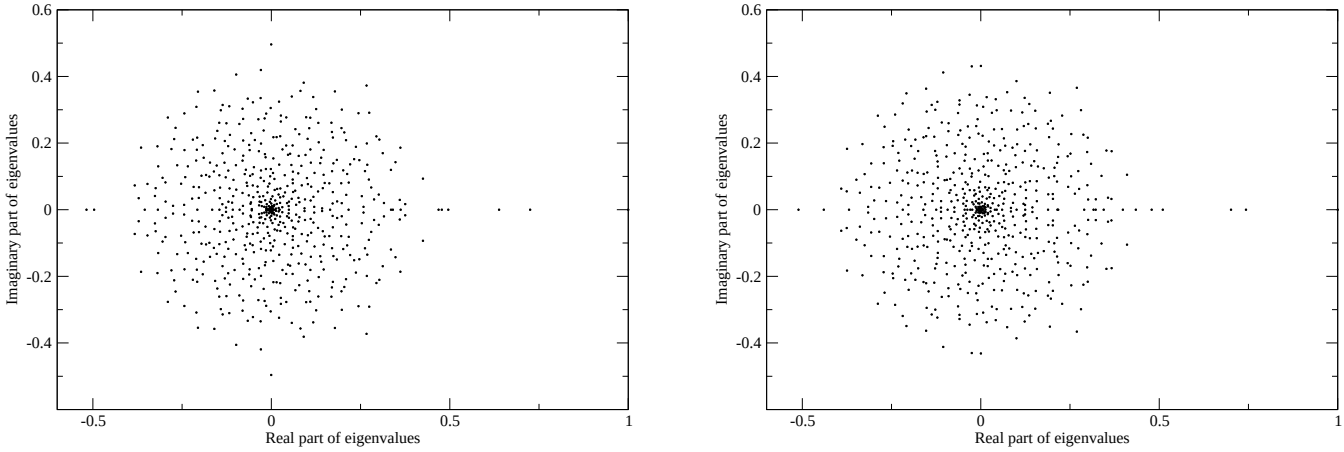


FIG. 22 – Spectres des valeurs propres dans le plan complexe pour G^* avec $\alpha = 1$, à gauche Gnugo (ordinateur) 9×9 non Monte-Carlo 20 000 parties et à droite Gnugo (ordinateur) 9×9 Monte-Carlo 20 000 parties, les zones denses en valeurs propres sont toutes les deux étalées.

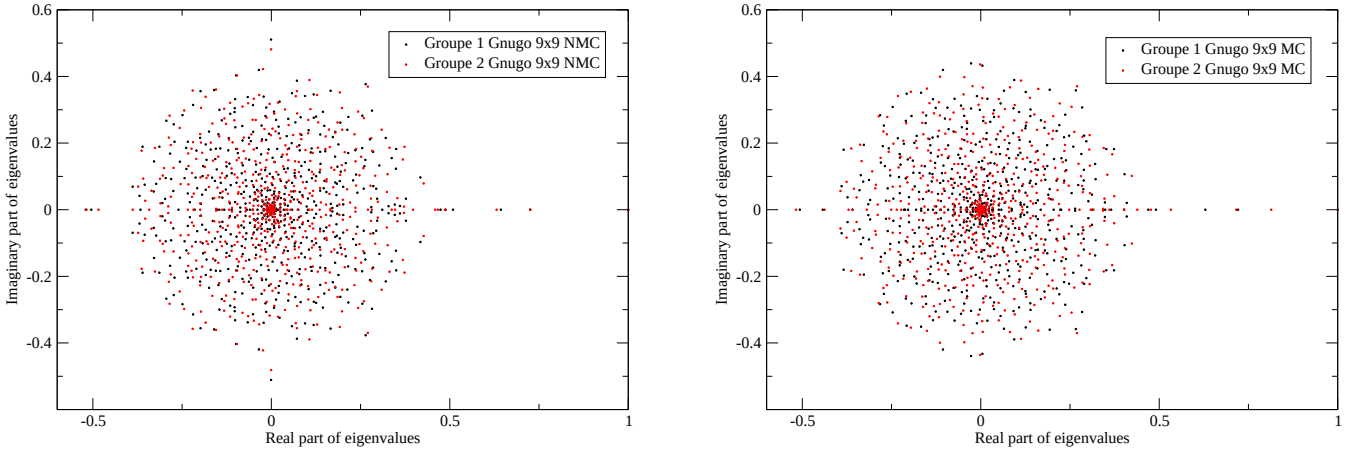


FIG. 23 – Spectres des valeurs propres dans le plan complexe pour G^* avec $\alpha = 1$, à gauche Gnugo (ordinateur) 9×9 non Monte-Carlo 2×10000 parties, à droite pour Gnugo (ordinateur) 9×9 Monte-Carlo 2×10000 parties, le nuage de point noir est relatif au groupe 1 et le nuage rouge au groupe 2 de chaque type d'algorithme. Les spectres sont étalées pour les quatre réseaux et on voit nettement que chaque groupe d'un même type d'algorithme leurs spectres coïncident, il n'y a pas de différences visibles entre Monte-Carlo et non Monte-Carlo ici.

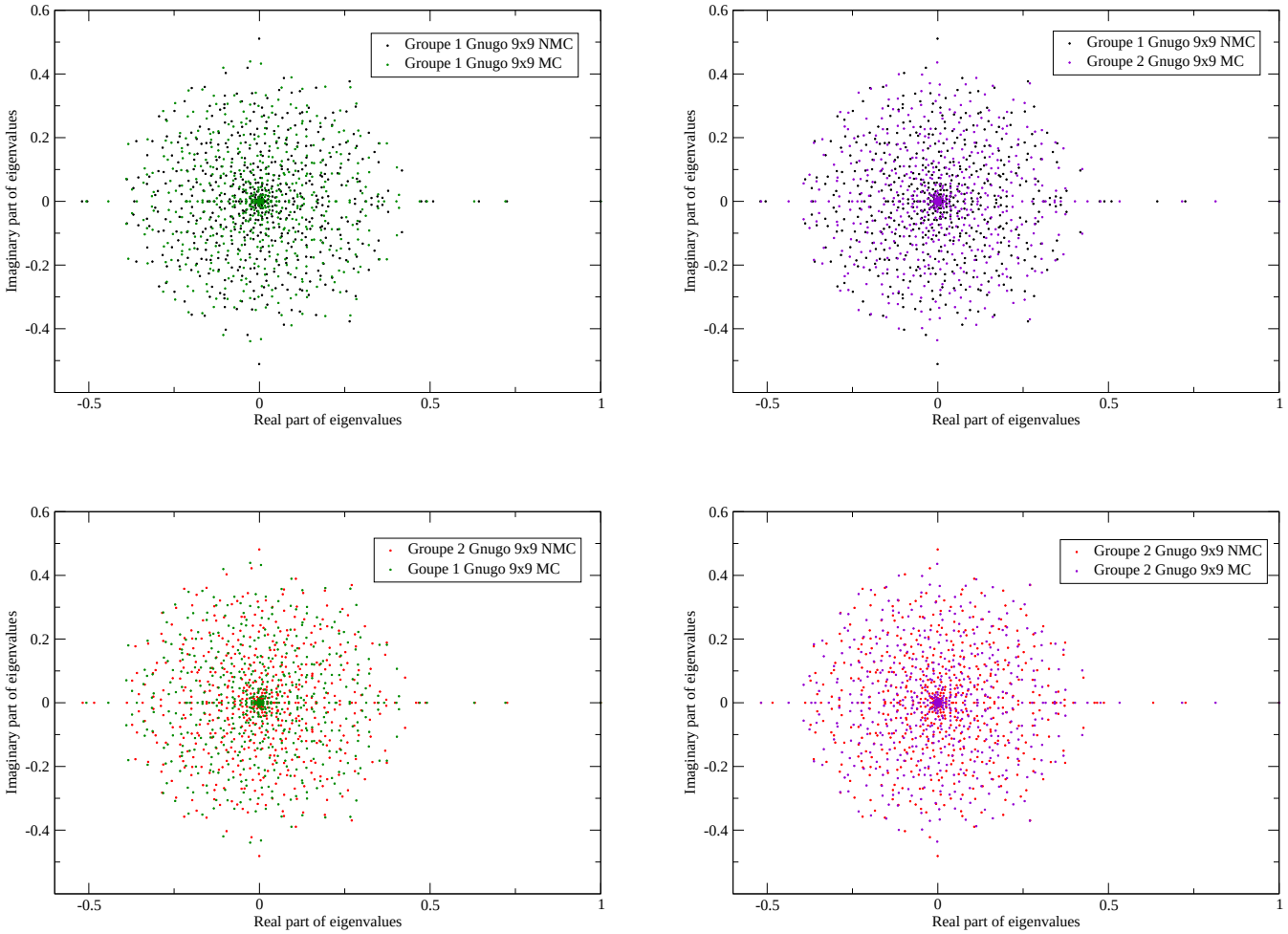


FIG. 24 – Spectres des valeurs propres dans le plan complexe pour Gnugo (ordinateur) 9×9 pour 4 sous groupes de 10 000 parties pour G^* avec $\alpha = 1$, en noir le groupe 1 non Monte-Carlo en vert le groupe 1 Monte-Carlo, en rouge le groupe non Monte-Carlo et en violet le groupe 2 Monte-Carlo. On voit avec ces quatre figures que les 4 réseaux ont leurs spectres qui coïncident, il n'y a pas de différences visibles entre types Monte-Carlo et non Monte-Carlo ici.

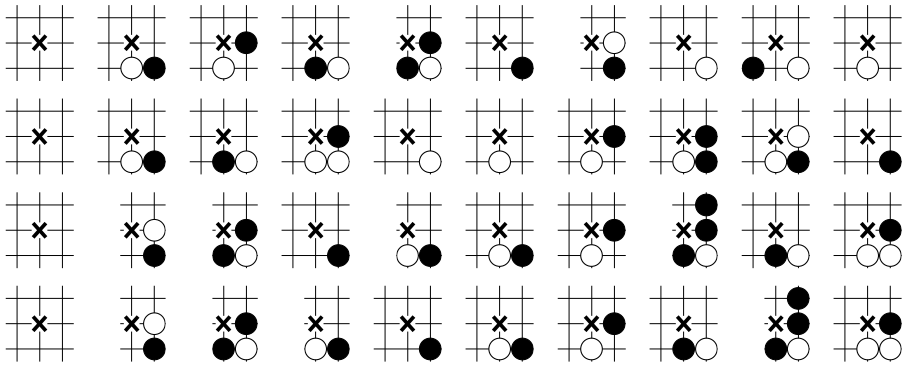


FIG. 25 – Top 10 des CheiRank avec le joueur noir jouant sur la croix, de haut en bas Gnugo (ordinateur) 19×19 , parties amateurs U-go (humain) 19×19, Gnugo 9×9 sans et avec option Monte-Carlo

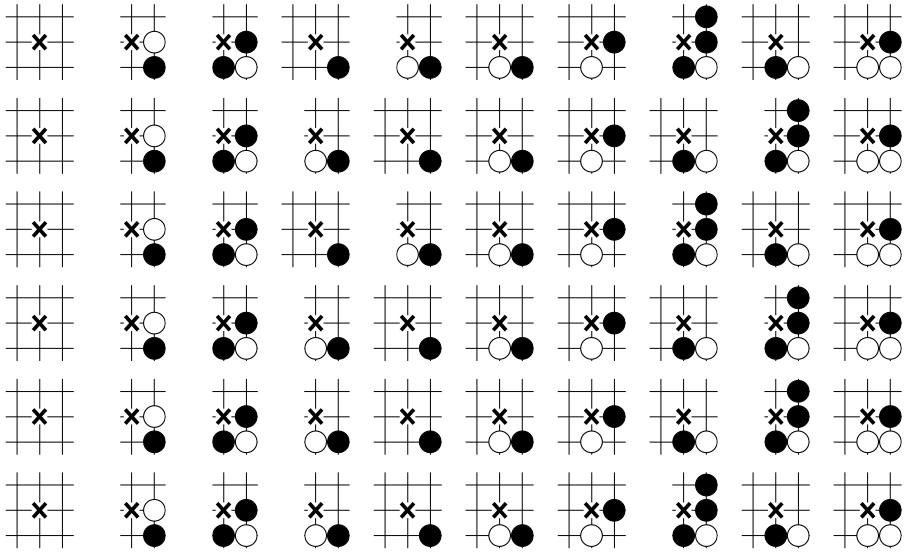


FIG. 26 – Top 10 des CheiRank avec le joueur noir jouant sur la croix, de haut en bas Gnugo (ordinateur) 9×9 20 000 parties sans Monte-Carlo, avec Monte-Carlo, 9×9 Groupe 1 de 10 000 parties, Groupe 2 de 10 000 parties et 9×9 option Monte-Carlo Groupe 1 et groupe 2 de 10 000 parties chacun

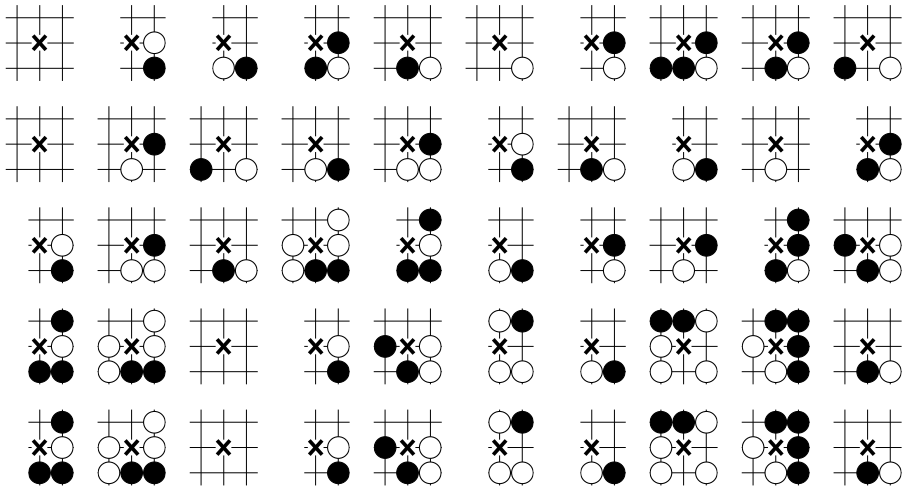


FIG. 27 – Les top 10 des plaquettes de 20 000 parties Gnugo (ordinateur) 9×9 non Monte-Carlo avec 5 autres valeurs propres, de haut en bas λ_2, λ_3, λ_4, λ_5 et λ_6

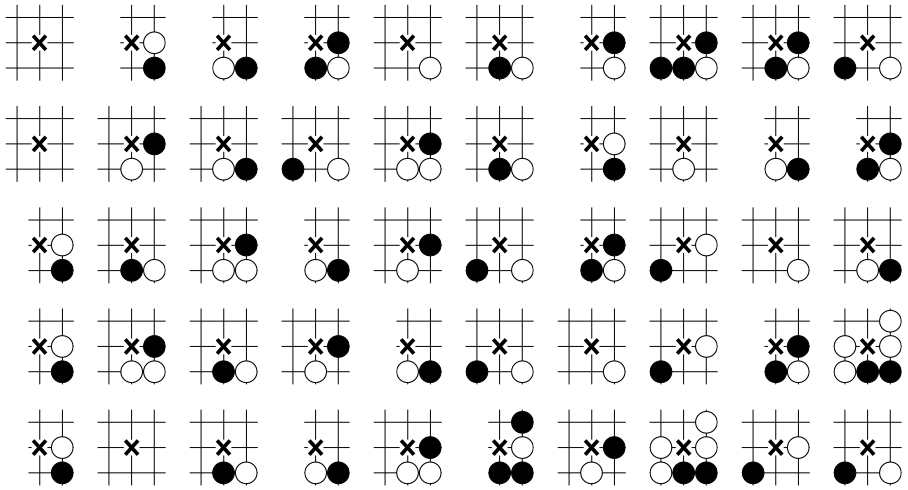


FIG. 28 – Les top 10 des plaquettes de 20 000 parties Gnugo (ordinateur) 9×9 Monte-Carlo avec 5 autres valeurs propres, de haut en bas λ_2, λ_3, λ_4, λ_5 et λ_6

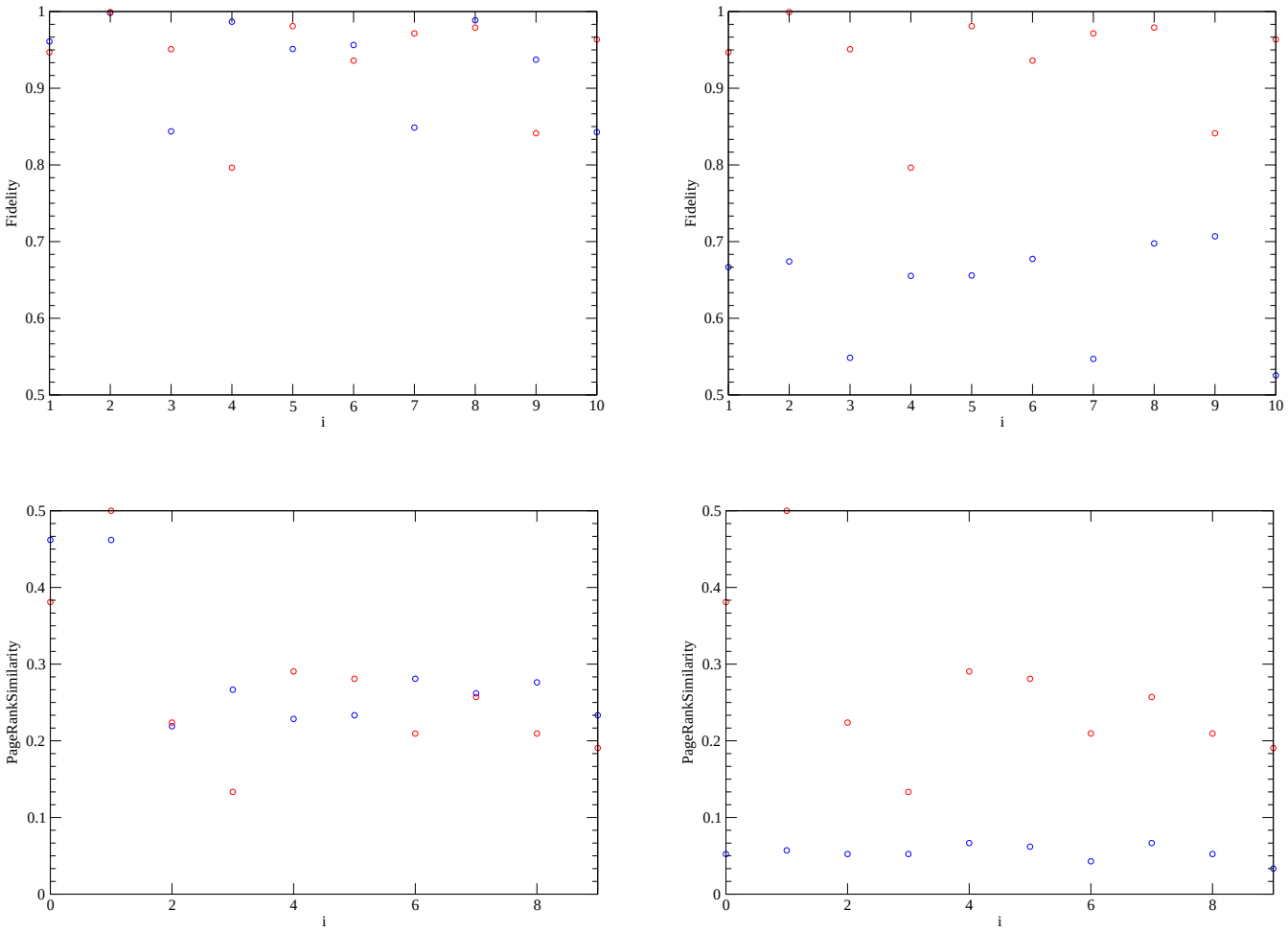


FIG. 29 – Tests de Turing avec Gnugo (ordinateur) et U-go (humain) 19×19 : les figures du haut sont pour la fidélité moyenne pour les 7 premiers vecteurs propres droits pour différentes banques de données, pour i = 1 et 2 nous avons 4000 parties , i>2 nous avons 1000 parties, à gauche en rouge U-go avec étalon U-go et en bleu Gnugo avec étalon Gnugo et à droite en rouge U-go avec étalon U-go et en bleu Gnugo avec étalon U-go. Les figures du bas sont pour la similarité moyenne des 7 premiers vecteurs propres droits classés. On voit nettement que humain et ordinateur se distinguent avec la fidélité moyenne, en revanche on ne distingue plus avec la similarité moyenne.

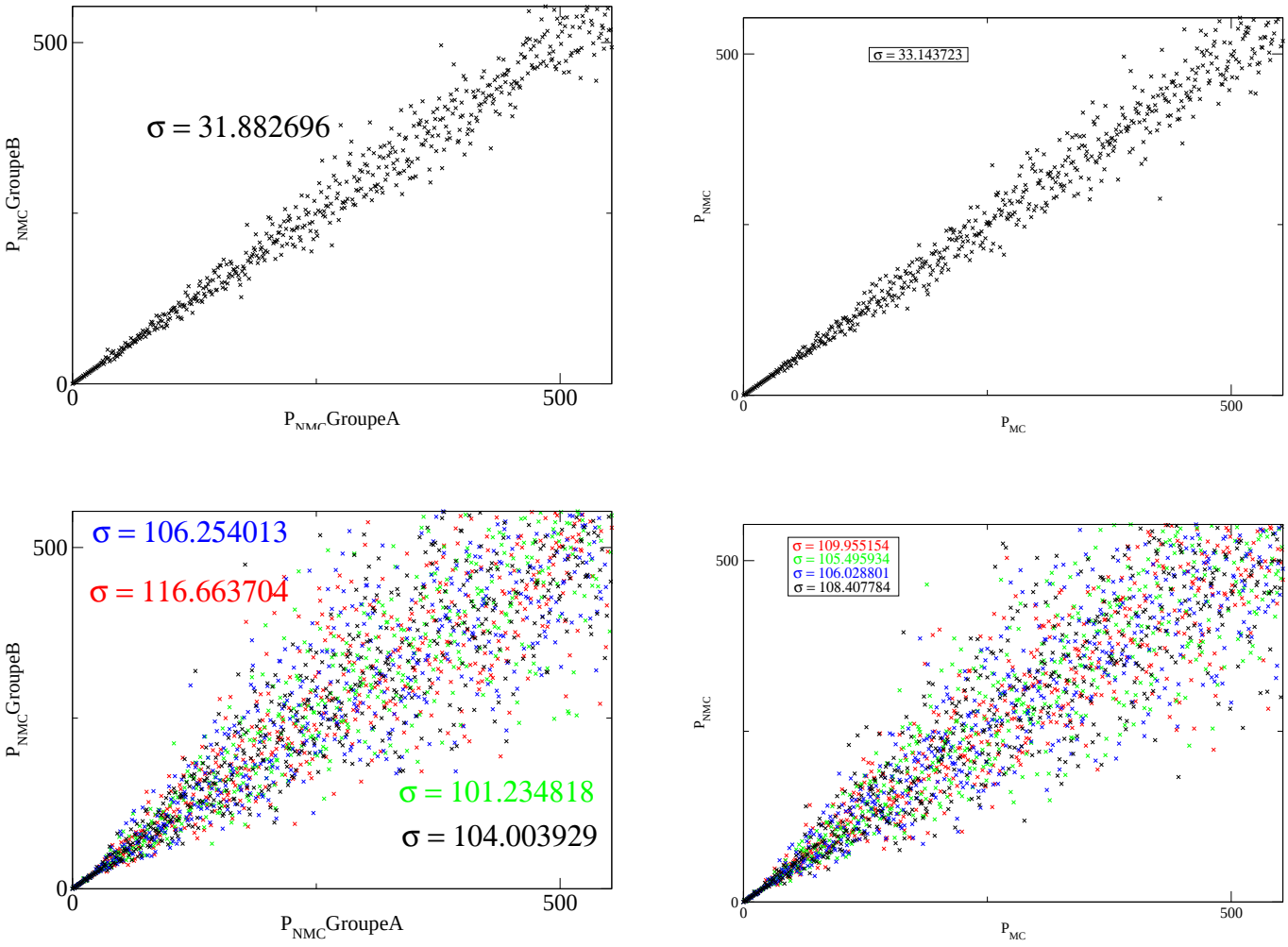


FIG. 30 – En haut corrélation entre PageRanks Gnugo (ordinateur) Monte-Carlo 9×9 pour deux groupes de même nombre de parties, ici 10 000 parties Monte-Carlo par groupe à gauche et 10 000 parties Monte-Carlo contre non Monte-Carlo à droite. En bas mêmes choses mais pour 4 couples de 1000 parties. On voit clairement que les nuages de points fuient la diagonales dans les figures du bas, les coefficients de corrélations changent considérablement avec un rapport supérieur à 2 en passant des figures du haut aux figures du bas on distingue donc les tailles des banques de données utilisées.