LE ROLE DES CONCEPTS SPATIAUX

DANS LA PROGRAMMATION DU JEU DE GO

Bruno Bouzy

C.R.I.P.5
UFR de mathématiques et d'informatique
Université René Descartes (Paris V)
45, rue des Saints-Pères 75270 Paris Cedex 06 FRANCE
tel: (33) (0)1 44 55 35 58 fax: (33) (0)1 44 55 35 35
e-mail: bouzy@math-info.univ-paris5.fr
http://www.math-info.univ-paris5.fr/~bouzy/

7 septembre 2001

1. INTRODUCTION

Le Raisonnement Spatial (RS) regroupe un ensemble de recherches effectuées sur des raisonnements s'appliquant sur des faits de nature spatiale. Par exemple, une application de description d'itinéraires s'intéressera, entre autres, à la formalisation spatiale d'un itinéraire; un système d'information géographique est une application du raisonnement spatial puisqu'il affiche à une échelle appropriée des informations relatives à des objets du monde réel situés dans l'espace. Un objectif de ces recherches est de trouver des formalismes de représentation de l'espace incluant des opérateurs adéquats.

Ayant travaillé dans le domaine de la Programmation du jeu de go (PdG), nous nous sommes aperçu que nos recherches avaient un point commun fort avec le domaine du RS. En effet, les chercheurs en programmation du jeu de go sont confrontés à l'élaboration d'une fonction d'évaluation. Cette fonction d'évaluation fait appel à de nombreux concepts, tels que l'« encerclement », le « territoire », le « regroupement », le « fractionnement », qui sont de nature spatiale et visuelle. Cet article présente donc une description des concepts spatiaux utilisé dans la programmation du Go.

Nous allons présenter brièvement un état de l'art du RS dans la partie 2, ensuite dans la partie 3, nous allons présenter celui de la PdG en précisant l'obstacle actuel principal de la PdG : la construction de la fonction d'évaluation. Dans la partie 4, nous décrirons les concepts spatiaux utilisés par la fonction d'évaluation et dans la partie 5 nous discuterons de ces concepts. Il n'est pas nécessaire de connaître le go pour lire cet article. Cependant une rapide présentation des règles figure en annexe.

2. RAISONNEMENT SPATIAL

Cette partie présente un résumé de l'état actuel des recherches en Raisonnement Spatial (RS). Un premier ensemble de recherches étudie les expressions spatiales présentes dans le langage naturel. Par exemple, « à côté de », « au dessus de », « derrière », « aller », « vers », « le long de », « dans » « autour », « jusqu'à » sont des expressions spatiales courantes. De nombreux travaux ont été effectués sur l'étude des expressions spatiales. [Herskovits 1982] est un travail fondateur sur les prépositions spatiales en anglais.

D'autres recherches importantes traitent du « raisonnement spatial qualitatif » (RSQ) dont le but est d'élaborer et utiliser des modèles formels « qualitatifs » de l'espace. L'idée du raisonnement spatial qualitatif est d'abstraire d'une représentation quantitative et continue des objets dont les propriétés

prennent des valeurs discrètes et dont les relations sont simples. Partant de cette abstraction qualitative, le système peut utiliser des lois de composition qualitatives entre les relations pour en déduire efficacement les relations entre les objets. Par exemple, si A, B et C sont trois objets tels que A et B sont liés par une relation et B et C par une autre, le problème est de connaître la relation éventuelle entre A et C. Si cette relation est trouvée grâce à une table de composition de relations, le système gagnera en performances. Cette idée suit celle d'Allen qui a obtenu un succès reconnu dans le raisonnement temporel [Allen 1983] où la dimension vaut 1. Pour le raisonnement spatial, il est plus difficile de faire des raisonnements qualitatifs car la dimension est 2 ou 3. [Cohn 1997] est une excellente revue de l'état de l'art en RSQ. [Hernandez 1994] propose une représentation qualitative de l'espace. Les études en RSQ font souvent des hypothèses différentes de celles faites en topologie générale. Par exemple, les systèmes formels ne reposent pas nécessairement sur la notion de point mais sur celle de région. De plus, alors que la topologie générale est basée sur la notion d'ouverture et de fermeture d'ensemble, les systèmes formels du RSQ peuvent ne pas faire de distinction entre une région, sa fermeture et son ouverture topologique. Le RSQ s'oppose au RS tout court, qui pourrait s'appeler RS quantitatif, où le raisonnement est fait à partir des données numériques du domaine. C'est pourquoi nous poserons la question de la nature qualitative ou quantitative du raisonnement au go dans la partie « Discussion » de cet article.

Le « calcul des connexions de régions » (RCC pour Region Connection Calculus) a été développé grâce aux travaux de Anthony Cohn et David Randell [Randell & al. 1992], [Cohn 1996]. RCC est basé sur le calcul de Clarke [Clarke 1981]; il utilise 8 relations basiques sur des régions spatiales : *DC* « est disjoint de », *EC* « est extérieurement connecté avec », *PO* « recouvre partiellement », *TPP* « est inclus et tangent à », *NTPP* « est inclus et non tangent à », *TPPI* « inverse de TPP », *NTPPI* « inverse of NTPP » et *EQUAL* « est égal à ». Ces relations sont définies à partir de la relation *C(x,y)*, qui se lit *`la région* x *est connectée avec la region* y'. La relation *C* est à la fois réflexive et symétrique et donne la valeur vraie dès que les deux régions *x* et *y* partagent un point commun. Des tables de composition peuvent être engendrées pour fournir une base du raisonnement spatial avec RCC. En lien avec [Clarke 1981], des modèles logiques ont été construits et démontrés complets et adéquats [Asher & Vieu 1995] en partant de travaux sur la méréologie [Lesniewski 1927].

Un certain nombre de méthodes du RSQ sont utilisées dans les Systèmes d'Informations Géographiques (SIG) pour créer des tables de composition de relations spatiales. Les méthodes xIM [Egenhofer 1989, 1991] sont des méthodes utilisant des intersections d'ensembles (IM pour Intersection Method). Etant donné un ensemble A, on note dA la frontière de A et A° son intérieur. Etant donnés alors deux ensembles A et B, la méthode 4IM demande quelle est la nature (vide ou non vide) des intersections d'ensembles A°, dA avec B°, dB. Elle classifie donc la disposition spatiale de deux objets en $2^4 = 16$ classes. La méthode 9IM [Egenhofer 1991] est analogue à la précédente avec l'utilisation supplémentaire du complémentaire A^{-1} d'un ensemble A. Elle obtient théoriquement $2^9 = 512$ classes. En réalité, 9 classes de relations existent dans le monde réel. Elles correspondent presque aux 8 classes de RCC8 : seule « Proper Overlap » de RCC8 correspond à deux classes réalistes, ces deux classes étant discriminées par 9IM. [Egenhofer 1991] construit la table de composition de relations spatiales et prouve sa validité en utilisant 9IM. Par exemple, cette méthode permet de déduire que si A « touche » B (A rencontre dB mais pas B) et si l'intérieur de B contient C, alors A et B sont disjoints. D'autres méthodes, toujours basées sur le même principe, font en plus intervenir d'autres propriétés d'ensembles et sont donc plus puissantes et moins restrictives. La méthode CBM (Calculus Based Method) [Clementini & Di Felice 1995] fait en plus intervenir la dimension des ensembles. Malheureusement toutes ces méthodes ne sont jamais satisfaisantes en pratique car les tables de compositions restent assez peu informatives. Par exemple, si A « touche » B et si B est recouvert par C, alors la méthode 9IM conclue que A « touche » C ou que A est dans l'intérieur de C ou que A est recouvert par C ou que A rencontre C, ce qui constitue une information assez faible. Dans le cas général, la méthode ne conclut pas sur une unique possibilité, ce qui serait utile en pratique. Par conséquent, les systèmes informatiques devant calculer correctement la disposition spatiale qualitative de deux objets connaissant leur dispositions qualitatives par rapport à un troisième ne peuvent pas utiliser les tables de composition dans la plupart des cas et doivent utiliser les données numériques sur les deux objets pour calculer l'information qualitative. Mais actuellement, le but des SIG est non seulement de fournir des informations géographiques de façon correcte, complète et performante mais aussi de les offrir sous une forme qui suit de près le sens commun de l'utilisateur. La « géographie naïve » est un champ d'études concerné par les modèles formels de la « géographie du sens commun » [Egenhofer 1995].

Le RS est une activité très importante des robots. Il serait trop long de citer toutes les publications de robotique concernant le RS. Nous ne citerons donc que le groupe QSIM de Benjamin Kuipers à l'université du Texas, reconnu mondialement [Kuipers & Byun 1988] pour les travaux sur le modèle « TOUR » [Kuipers 1978] et sur la « hiérarchie de sémantique spatiale » (SSH pour Spatial Semantic Hierarchy). L'idée de la SSH est la construction d'une « carte cognitive » qui représente la connaissance à grande échelle de l'environnement d'un agent (homme ou robot). La carte cognitive est ancrée sur l'interaction sensori-motrice de l'agent avec le monde et est centrée autour d'une description topologique. La carte est constituée de quatre niveaux : sensori-moteur, procédural, topologique et géométrique. Chaque niveau possède sa propre ontologie et ses propres méthodes de résolution de problèmes. L'idée fondamentale de la SSH est de rendre la connaissance spatiale de l'agent indépendante de son appareil sensori-moteur et de l'environnement dans lequel il se déplace.

Selon nous, deux points d'ordre méthodologique sont essentiels pour enrichir l'état des recherches en RS. D'abord, le RS de l'être humain n'étant pas une activité nécessairement verbale, nous pensons qu'il est utile de poursuivre les études sur des domaines *non verbaux.* Ensuite, le monde réel étant très complexe et les modèles théoriques de l'espace étant difficilement utilisables dans des applications informatiques réelles, nous pensons que les études sur le RS doivent porter sur des domaines toujours assez complexes mais surtout significativement *plus simples* que le monde réel. Nous pensons alors à des domaines tels que les jeux de réflexion. Qui plus est, les jeux de réflexion utilisant des damiers finis permettent de dépasser les problèmes liés à l'opposition entre la modélisation d'espaces continus et celle d'espaces discrets [Egenhofer & Sharma 93], [Galton 99], [Cohn & Varzi 99]. Le jeu de go par exemple est un domaine répondant à ces deux critères. Nous pensons donc que les études sur la programmation du jeu de go sont susceptibles d'enrichir les recherches en RS.

3. PROGRAMMATION DU JEU DE GO

Cette partie présente l'état de l'art de la programmation du jeu de go. Tout d'abord, elle rappelle la complexité combinatoire du jeu de go par rapport aux autres jeux à deux joueurs à information complète et à somme nulle, ensuite elle donne un historique de l'évolution du domaine et les résultats obtenus, enfin elle donne les caractéristiques générales des programmes de go suivant les concepts classiques de fonction d'évaluation, génération de coups et recherche arborescente. Les meilleures publications concernant l'état de l'art de ce domaine sont [Mueller 1998, 2000b].

3.1. Aperçu de la complexité du go

Ce paragraphe donne une idée de la complexité du jeu de go en la comparant à celles d'autres jeux à deux joueurs à somme nulle à information complète, comme les échecs ou othello.

Tout d'abord, il est important de signaler que la théorie de la complexité, discipline de l'informatique théorique, ne permet pas de départager les jeux à deux joueurs à somme nulle à information complète. En effet, d'une part le go, les échecs ou othello appartiennent à la même classe de complexité : le problème de déterminer une stratégie gagnante à partir d'une position quelconque est un problème « exponentiel en temps » en fonction de la taille du damier [Fraenkel & Lichtenstein 1981] [Robson 1982]. D'autre part, les joueurs jouent à ces jeux sur des damiers de taille fixe (8x8 aux échecs et à othello, 9x9, 13x13 ou 19x19 au go) et les programmeurs de ces jeux ne se soucient pas (est-ce un tort ?) de ce que donneraient leurs algorithmes sur des damiers de taille beaucoup plus grande que la taille des damiers usuels.

Pour différencier le jeu de go des échecs ou d'othello, nous préférons effectuer un calcul numérique approximatif avec quelques paramètres pertinents des jeux : B, le facteur de branchement moyen, L, la longueur moyenne d'une partie, A, la taille estimée de l'arbre du jeu ($= B^L$).

Jeu à 2 joueurs	B	L	A
othello	10	60	10^{60}
échecs	35	80	10^{120}
go	200	300	10^{700}

Tableau 1

Le tableau 1 montre que si l'on construisait l'arbre du jeu de go, sa taille serait 10^{580} fois plus grande que celle de l'arbre des échecs. Au go, il n'est donc pas envisageable d'utiliser la « force brute » basée sur la recherche arborescente, les bases d'ouvertures et de finales, qui a si bien réussi à la programmation des échecs [Hsu & al. 1990]: Deep Blue, programme et machine d'IBM descendant de Deep Thought, a battu Kasparov, le champion du monde aux échecs en 1997.

Evidemment, le lecteur pourra remarquer que ce calcul suppose que la taille du damier de go est 19x19. Avec un damier 9x9, on trouverait une complexité égale à celle des échecs. Pourtant, même sur 9x9, la machine reste d'un niveau très médiocre. Cela veut dire que la complexité numérique donnée par le tableau ci-dessus ne reflète pas complètement la complexité du go, qui serait alors d'une autre nature... Pour l'instant retenons que la complexité du go est numériquement plus grande que celle des échecs ou d'othello et que les méthodes de seules recherches arborescentes sont efficaces aux échecs ou à othello [Buro 1994] et inefficaces au jeu de go.

3.2. Bref historique et niveau atteint sur la partie complète

Ce paragraphe dresse un bref historique de la PdG. Il se distingue du paragraphe suivant par la complétude du problème abordé : alors qu'au paragraphe suivant on s'intéressera à certains sous-problèmes posés par le jeu de go, ici on s'intéresse au problème de jouer le mieux possible une partie de go complète sur des damiers 19x19. Pour comprendre le niveau des programmes actuels, il est nécessaire de rappeler qu'il existe une échelle basée sur un système de « kyu » et de « dan » : $10^{ème}$ dan pro correspond au niveau de champion du monde professionnel, 1^{er} dan pro et $7^{ème}$ dan amateur à celui de champion d'Europe, 1^{er} dan amateur ou 1^{er} kyu à celui de bon joueur, $5^{ème}$ à $10^{ème}$ kyu à celui de joueur moyen, $10^{ème}$ à $20^{ème}$ kyu à celui de joueur débutant et $30^{ème}$ à $50^{ème}$ kyu à celui de novice complet.

Le début des recherches sur la programmation du go remonte aux années soixante. Les premiers résultats théoriques sur des damiers de taille réduite ont été trouvés par [Thorp & Walden 1972]. Albert Zobrist fut le premier créateur de méthodes ou algorithmes encore utilisés actuellement dans les programmes de go : son modèle d'influence [Zobrist 1969] est l'exemple le plus célèbre. Bruce Wilcox a écrit de nombreux programmes de go dont le niveau à l'époque devait être $25^{ème}$ kyu [Wilcox 1979]. De nombreux travaux ont été faits sur la manipulation des connaissances par un programme, connaissances déclaratives [Mano 1984], affinement de connaissances [Shirayanagi 1989]. D'autres portent sur le lien entre la théorie des jeux de Conway [Conway 1976] et la programmation du go [Mueller 1993, 1995, 1999]. Un sous-problème crucial du jeu de go complet est la résolution des problèmes de « vie et mort des groupes ». Ces problèmes surviennent dès qu'un groupe de pierres est encerclé[1]. Des formalisations de ce problème ont été faites mathématiquement [Benson 1976] ou expérimentalement intégrées dans des programmes complets [Kraszek 1988] [Chen & Chen 1999] ou spécifiques [Wolf 2000]. Des programmes intègrent des méthodes d'apprentissage avec des réseaux de neurones [Enzenberger 1996] ou bien par acquisition automatique de règles [Cazenave 1996]. Au travers de la littérature académique, l'ensemble de ces travaux peut apparaître assez hétérogène et sans grande unité. Il faut bien préciser que la programmation du jeu de go en est encore à ses débuts et que les techniques publiées ne sont pas encore unifiées. Il est trop tôt pour dire qu'il existe un modèle standard sur lequel la communauté soit d'accord. Tout au moins, si le modèle standard existe, il n'est pas publié. En effet, les meilleurs programmes sont commerciaux et leurs techniques sont, sauf exception [Chen & Chen 99], secrètes.

[1] La vie et la mort des groupes sera présentée au paragraphe 4.7

Les premières compétitions internationales remontent à 1987 avec l'apparition de nombreux programmes dont nous évaluons le niveau approximatif à 20ème kyu. Goliath de Mark Boon [Boon 1991], 10ème kyu à l'époque, fut le premier programme à sortir du lot en gagnant trois fois le championnat du monde des programmes en 1989, 90 et 91. Depuis, Handtalk a gagné quatre fois le championnat du monde et sa force est environ 7ème kyu. Malheureusement ce programme, qui s'appelle désormais Goemate et qui est écrit en assembleur, n'est que très peu décrit. De plus, cette description n'est pas informatique du tout mais uniquement liée à la technique du go proprement dite [Chen & Chen 99]. En novembre 2000, Wulu a gagné la compétition. C'est un programme produit par la compagnie du professeur Chen, auteur de Goemate. En tant que vainqueur, Wulu a ensuite rencontré trois jeunes apprentis professionnels, environ 5ème dan amateur, avec neuf pierres d'avance mais il a perdu les trois parties. Le dernier challenge réussi date de 1997 quand Handtalk a gagné avec onze pierres d'avance.

On a vu que la recherche arborescente seule est une approche interdite par des arguments numériques et combinatoires. Une autre approche, celle de l'IA, consiste à dire que puisque l'homme est très supérieur à la machine, il est tentant de programmer le go en simulant l'activité des joueurs humains. Mais alors, on se heurte au problème d'extraction des connaissances humaines. Ce problème est surtout dû à la nature implicite des connaissances pour jouer au go. En effet, la majorité des connaissances sur le jeu de go sont visuelles : connaissances pour regrouper, pour encercler, pour séparer, etc. Il est alors très difficile de trouver les connaissances élémentaires qui sont à la base du raisonnement des joueurs humains. Enfin, une partie de l'explication relève de l'informatique. Une idée identifiée par le processus d'extraction des connaissances doit ensuite être validée expérimentalement en écrivant un programme informatique la simulant. La mise au point du programme de simulation, les informaticiens le savent, n'est pas immédiate.

3.3. Résultats obtenus sur des sous-problèmes spatio-temporels

Il est important de savoir que sur des problèmes de « vie et de mort », terminologie des joueurs de go, le programme GoTools est excellent et possède un niveau équivalent à 5ème dan amateur [Wolf 1994, 2000]. Les problèmes de vie et de mort sont des problèmes qui arrivent lorsqu'un groupe de pierres est complètement encerclé. Il faut alors déterminer si le groupe est « mort » ou « vivant » et où jouer.

De même, sur des positions de fin de partie, en fait sur des positions ad hoc, le modèle mathématique de Elwin Berlekamp donne des résultats meilleurs que ceux donnés par des professionnels [Berlekamp 1991] [Berlekamp & Wolfe 1994]. Ce modèle est basé sur la théorie des jeux de Conway [Conway 1976] [Conway & al. 1982], théorie qui s'applique aux jeux sommables et indépendants. Il faut noter que la différence de résultat entre le modèle de Berlekamp et le résultat de joueurs professionnels n'est que d'un point au plus; le joueur de go appréciera. Il faut noter aussi que les positions testées ne correspondent pas à des positions de parties réelles ; ces positions sont pré-traitées et contiennent l'information sur la « vie » et la « mort » des groupes.

Ces deux résultats excellents sur des versions réduites spatialement (les problèmes de « vie et de mort ») ou temporellement (les problèmes de fin de partie) du jeu de go complet doivent être pris comme des signes encourageants pour obtenir des résultats analogues sur le jeu de go complet. Tout le problème est maintenant de généraliser ces approches spécifiques.

3.4. Caractéristiques des programmes

Ce paragraphe donne un aperçu des caractéristiques des programmes de go [Fotland 1992, 1996] au travers du paradigme classique des jeux : la fonction d'évaluation, la génération de coups et la recherche arborescente.

La fonction d'évaluation (FE) est la partie la plus complexe d'un programme de go et aussi la partie la plus intéressante pour le RS. En effet, elle décrit l'occupation spatiale du damier. On peut définir une fonction *concrète* de manière triviale mais il est plus difficile de définir des fonctions *abstraites* représentatives du go, adéquates et calculables rapidement. La FE *concrète* est la fonction d'évaluation la plus simple que l'on puisse imaginer : on donne +1 pour une intersection noire et –1 pour une

intersection blanche. Elle est inutilisable telle quelle car cette fonction n'a de signification que sur les positions terminales où plus aucune capture n'est possible, c'est-à-dire à une profondeur 720 environ (2 fois la taille du goban environ, l'expérience le prouve) par rapport au début de la partie. On retrouve l'argument combinatoire pour interdire cette approche. Le joueur humain utilise des concepts tels que, par exemple, «groupe», «territoire», «fraction» etc. Si l'on arrive à identifier et simuler ces concepts, on peut alors définir une FE *abstraite*. La description détaillée d'une fonction d'évaluation abstraite est l'objet de la prochaine partie.

La génération de coups (GC) est essentiellement faite en utilisant du « pattern-matching ». Le pattern-matching permet d'apparier une partie de position avec des formes contenues dans une base de formes. Une forme conseille un ou plusieurs coups. Actuellement [Boon 1989], le pattern-matching est concret : il s'applique à des dispositions spatiales de pierres sans aucune abstraction. Une caractéristique de la GC est son couplage avec la FE. Pour un objet identifié par la FE, correspondent des objectifs à atteindre sur cet objet. On peut donc effectuer la GC suivant l'un de ces objectifs. Une autre caractéristique de la GC découle de la décomposition d'une position globale en positions locales. Pour étudier une position locale, on doit considérer qu'un joueur peut *jouer plusieurs coups de suite* dans cette position ou au contraire *passer*. Enfin, une troisième caractéristique de la GC est d'être *sélective* : étant donné le nombre de coups très grand par position, la GC peut engendrer sélectivement les coups au voisinage du dernier coup, ce qui correspond bien à l'idée que l'on se fait de l'étude d'une position locale.

La recherche arborescente (RA) au go possède également quelques spécificités [Mueller 2000a]. D'abord, la nature des algorithmes de recherche arborescente est dans l'ensemble de type « alpha-beta ». Mais « pn-search » [Allis 1994] est une méthode intéressante lorsque la FE se calcule lentement et prend ses valeurs dans {0, 1}, ce qui est le cas de la RA *sur objectif*. Ensuite, autre caractéristique, les objectifs peuvent n'être atteints qu'en plusieurs coups : ainsi une RA peut être utilisée non pas pour trouver le meilleur coup mais pour savoir combien de coups de la même couleur sont nécessaires pour atteindre un objectif et donner une importance stratégique à l'objectif. Une autre caractéristique de la RA découle du fait qu'une position locale peut être délaissée par les deux joueurs et que, plus tard, l'un ou l'autre des deux joueurs y joue en premier. Pour une position locale, cela oblige donc le programme à effectuer *deux recherches arborescentes*, chacune supposant que l'un ou l'autre des deux joueurs a le trait. On doit ensuite stocker deux informations et les réutiliser ensuite. Concernant la profondeur des RA, on peut dire que les programmes actuels n'effectuent généralement pas de RA sur FE globale à plus d'une profondeur un ou deux. La RA globale est limitée car elle ne peut conclure que sur des positions calmes, peu nombreuses au long de la partie [Chen 2000]. Sur un objectif local, la profondeur des RA est faible, quelques coups environ. Dans le cas d'une RA tactique sur la prise d'une chaîne, la profondeur peut être grande car les programmeurs arrivent à ramener la génération de coups à un seul coup dans la plupart des noeuds de l'arbre et à quelques coups sur les autres noeuds. Récemment, des techniques originales de RA, decomposition-search [Mueller 2000a] et lambda-search [Thomsen 2000], ont été développées et appliquées au go.

4. CONCEPTS SPATIAUX DE LA FONCTION D'EVALUATION

Cette partie montre ce qu'est une FE au go. Bien qu'il existe une multitude de manières de construire une FE, nous pensons que de nombreux concepts relatifs à la représentation de l'espace doivent nécessairement intervenir. C'est la raison de cet article. Cette partie montre donc l'ensemble des concepts spatiaux utilisables par une FE et, par là, montre que l'élaboration de la FE relève du RS.

La démarche suivie par les programmeurs de go étant expérimentale, les concepts présents dans les FE des PdG possèdent évidemment un outil associé de simulation sur ordinateur et sont donc, en ce sens, des concepts computationnels. Pour chaque concept computationnel, nous donnons les outils mathématiques associés à la construction d'un objet, exemple de ce concept computationnel, créé sur une position donnée. Cette vue pragmatique ne doit pas faire oublier au lecteur l'autre partie du travail de recueil d'expertise auprès des joueurs humains. En effet, chaque concept computationnel présenté ici correspond à un ou plusieurs concepts cognitifs. Ces concepts cognitifs sont identifiés grâce à leur expression dans les commentaires des parties de go par des termes tels que « groupe », «fraction», «encerclement», «intérieur», «extérieur», «territoire», «interaction », «inversion», «symétrie», «rotation», «agrégation », «incrémentalité». Rien qu'à la lecture de la liste de ces termes, nous sentons

immédiatement la nature spatiale d'une FE. Pour illustrer notre description de FE, nous prenons l'exemple de la position de la figure 1.

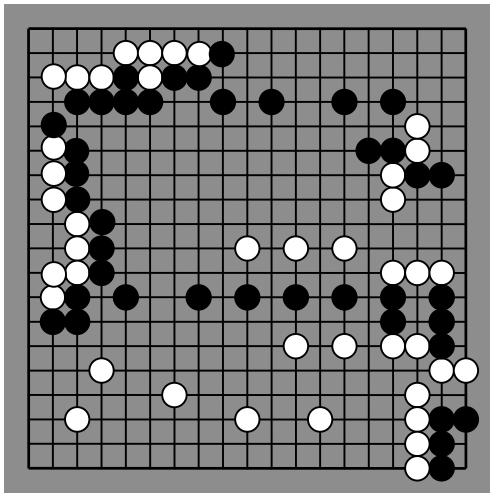

Figure 1

Nous avons choisi de présenter la liste des concepts de manière ascendante : nous partons des intersections du damier et nous construisons des concepts abstraits jusqu'à obtenir la FE en entier. Pour cela, nous présentons d'abord des concepts *géométriques* tels que l'«inversion», la «symétrie», la «rotation», puis nous décrivons des concepts *topologiques* tels que «groupe-connecté » et « fraction ». Ensuite nous relions « groupe-connecté » et « fraction » par la notion très importante d'« encerclement ». Après, nous passons à l'étude *morphologique* [Serra 1982] de la position avec les concepts de « territoire », « influence » et « groupe-morphologique » et deux concepts intuitifs tels que « intérieur »[1] et « extérieur ». Ensuite, nous décrivons les concepts purement *qualitatifs*, non spatiaux cette fois, de « vie » et de « mort » avec à nouveau les concepts d'«inversion» et d'«agrégation». Enfin, nous terminons par un résumé de la FE abstraite, de l'incrémentalité et de l'aspect bi-échelle.

4.1. « inversion », « rotation », « symétrie »

Au service de la GC ou de la FE, les programmes utilisent des bases de formes. Une forme est une disposition spatiale de pierres possédant certaines propriétés. L'algorithme de pattern-matching d'un programme balaye la base de formes et les compare à des parties de la position réelle. Lorsque la comparaison réussit, l'information stockée avec la forme est utilisée par la FE ou la GC. Le point important du point de vue spatial est que la comparaison se fait modulo une inversion de couleur Noir-Blanc, une rotation de 0°, 90°, 180° ou 270° et/ou une symétrie par rapport à un axe, vertical par exemple. Ainsi, un appariement d'une forme avec un partie de position réelle comprend en fait 2x4x2 = 16 comparaisons effectives [Boon 1989]. Ces concepts d' « inversion », de « rotation » et de « symétrie » sont à relier directement avec ce que fait, entre autres choses, l'esquisse primaire de l'appareil visuel humain [Marr 1982]. La suite de cette partie montre comment les formes reconnues servent à initialiser la construction de la FE. Nous allons considérer des concepts topologiques tels que la connexion, la fraction et l'encerclement.

4.2. « groupe-connecté »

Dans ce paragraphe, l'objectif est de définir des « groupes-connectés ». En effet, la règle du jeu (cf. annexe) définit des chaînes de pierres comme des ensembles 4-connexes d'intersections de même couleur, mais le joueur de go utilise surtout les « groupes-connectés » comme base de son raisonnement. Le but de ce paragraphe est de montrer comment un « groupe-connecté » est construit.

Etant données deux chaînes voisines et de même couleur, on effectue deux recherches arborescentes (une recherche si Noir commence et une recherche si Blanc commence) pour déterminer si ces chaînes

[1] « Intérieur » sera défini avec l'aide de la morphologie mathématique [Serra 1982] et, en ce sens, n'est pas classé comme un concept topologique.

sont virtuellement connectées. Chacune de ces recherches sont des recherches arborescentes de type alpha-bêta contenant plusieurs séquences de coups prouvant une connexion virtuelle. Nous employons le terme de connexion virtuelle pour se différencier de la 4-connexion de la règle du jeu. Bien sûr, 4-connexion entraîne connexion virtuelle et non l'inverse. Lorsque deux chaînes sont virtuellement connectées, elles font alors partie du même « groupe-connecté ».

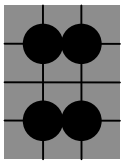

Figure 2

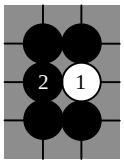

Figure 3

Les deux chaînes noires de la figure 2 sont virtuellement connectées car même si Blanc joue le premier (cf. figure 3), Noir peut connecter effectivement. (Si Noir joue le premier, les deux chaînes sont immédiatement connectées.) On dit donc que le dessin de gauche est un « connecteur ». On notera connecteur '>' pour exprimer que l'issue, positive, est indépendante du trait [Conway 1976].

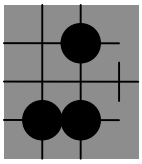

Figure 4

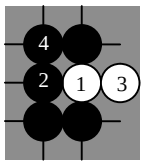

Figure 5

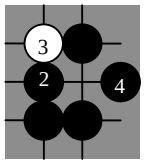

Figure 6

Les deux chaînes noires de la figure 4 sont virtuellement connectées. Si Blanc joue le premier, Noir est connecté après Noir 4 (figure 5). Si Blanc joue 3 sur la figure 6, alors Noir 4 capture la pierre blanche 1 et Noir est connecté (car Blanc ne peut plus rejouer en 1). La figure 4 est donc un autre exemple de connecteur '>'.

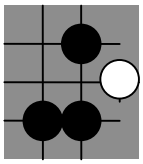

Figure 7

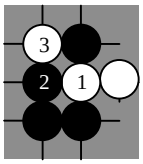

Figure 8

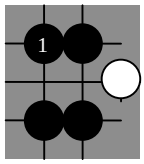

Figure 9

Sur la figure 7, on suppose que la pierre blanche du dessin de gauche possède suffisamment de libertés. Alors, la figure 7 n'est pas un connecteur au sens précédent car si Blanc joue le premier (figure 8), Noir 2 bloque mais Blanc 3 coupe et les chaînes noires sont déconnectées effectivement. Si Noir joue le premier (figure 9), les chaînes noires sont connectées effectivement. Dans ce cas (l'état du connecteur dépend du trait), on dit que le connecteur est '*' [Conway 1976].

Il faut noter que la notion de connecteur est locale. Même si les pierres extrémités d'un connecteur font globalement partie de la même chaîne (c'est possible si la chaîne fait le tour du dessin), elles peuvent ne pas être 4-connexes localement.

Les « groupe-connectés » sont alors définis comme les groupes de chaînes virtuellement connectées, c'est-à-dire reliés par des « connecteurs » '>'. Sur la position exemple de la figure 1, un programme obtient entre autre le « groupe-connecté » de pierres de la figure 10.

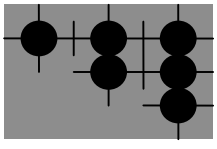

Figure 10

Nous discuterons l'adéquation de cette construction qualitative des « groupe-connectés », basée sur la transitivité de la relation « fait-partie-du-même-groupe-connecté » reliant deux groupes-connectés en

cours de construction, dans la partie 5 de cet article. Nous y verrons que cette construction n'est pas toujours correcte. Le concept topologique dual de la connexion étant celui de la séparation et de la fraction, voyons maintenant ce que donne ce concept.

4.3. « fraction »

Dans ce paragraphe, l'objectif est de définir des « fractions ». L'idée du concept de « fraction » s'exprime dans le jargon du go en disant que deux pierres non connectées peuvent tout de même séparer l'adversaire. Dans le vocabulaire mathématique, c'est l'idée du théorème de Jordan qui dit qu'une courbe fermée 8-connexe sépare l'espace en deux composantes 4-connexes. En suivant le paradigme ascendant-descendant, l'idée du concept de « fraction » consiste à fractionner l'ensemble du damier en « fractions » (idée descendante) alors que l'idée du concept de « groupe-connecté » était de regrouper des intersections (idée ascendante). Peu de travaux existent concernant la notion de fraction en RS. Cependant, on peut relier notre définition de fraction à celle de partition en RS [Erwig & Schneider 1997].

Pour définir une « fraction », il est nécessaire de définir des « séparateurs ». Etant données deux chaînes voisines et de même couleur, on effectue à nouveau deux recherches arborescentes (une recherche pour chaque couleur) afin de déterminer si ces chaînes séparent l'adversaire. Lorsque deux chaînes possèderont une fonction de séparation, elles feront alors partie du même « séparateur ». Une « fraction » sera ensuite définie comme un ensemble connexe dont les frontières sont constituées des « séparateurs »[1].

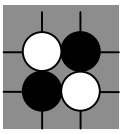

Figure 11

Sur la figure 11, les deux pierres noires ne sont pas 4-connexes. Cependant elles « séparent » les pierres blanches car elles sont 8-connexes. On définit donc un « séparateur » comme un ensemble de chaînes dont on ne peut supprimer leur fonction de séparation.

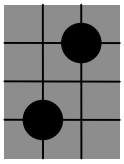

Figure 12

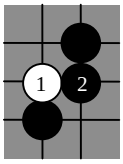

Figure 13

Les deux pierres noires de la figure 12 « séparent » car même si Blanc joue le premier (figure 13), les pierres noires sont 8-connexes et séparent la gauche de l'espace de la droite. A fortiori si Noir joue le premier.

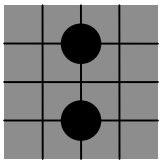

Figure 14

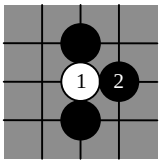

Figure 15

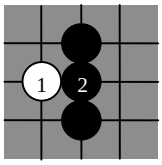

Figure 16

Sur la figure 14, les deux pierres noires du dessin de gauche « séparent » la gauche de l'espace de la droite. Le fait que cette figure soit un séparateur est prouvé par les séquences des figures 15 et 16. En effet, à la fin de ces deux séquences, les pierres noires sont 8-connexes. A nouveau, on notera ces « séparateurs » avec '>' [Conway 1976].

[1] Une fraction n'est pas un ensemble connexe de séparateurs mais bien un ensemble connexe *dont la frontière* est constituée d'un ensemble de séparateurs.

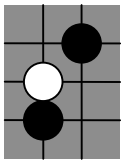

Figure 17

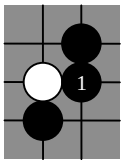

Figure 18

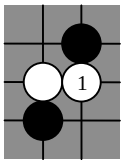

Figure 19

Sur la figure 17, les deux chaînes noires ne sépareront la gauche de la droite que si Noir joue en premier (figure 18) . Si Blanc joue en premier (figure 19), les deux pierres noires ne peuvent plus être 8-connexes et elles perdent leur fonction de séparation. Dans ce cas, on dit que le « séparateur » est '*' [Conway 1976].

Il faut noter ici encore que la notion de séparateur est locale. Même si les pierres extrémités d'un séparateur font globalement partie de la même chaîne 4-connexe ou 8-connexe (c'est possible si la chaîne fait le tour du dessin), elles peuvent ne pas être 8-connexes localement.

Les « séparateurs » '>' (respectivement '*') permettent de définir les « fractions » d'une couleur donnée comme étant les ensembles connexes maximaux dont les frontières sont des « séparateurs » '>' (respectivement '*') de cette couleur.

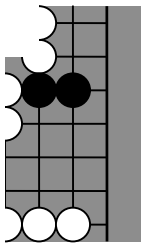

Figure 20

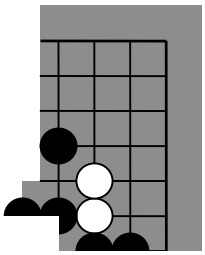

Figure 21

La figure 20 représente une « fraction » blanche de type '>' de la figure 1. Elle est blanche car limitée par des séparateurs '>' blancs. La figure 21 représente une « fraction » noire de type '>' de la figure 1. Nous donnerons d'autres exemples de fractions dans le paragraphe suivant à propos de l' « encerclement ».

4.4. « encerclement »

Il est important de remarquer qu'un « groupe-connecté » d'une couleur est toujours inclus dans une « fraction » de l'autre couleur. Et qu'une fraction d'une couleur contient zéro, un ou plusieurs groupes de l'autre couleur. On peut donc se demander quelle est la nature de la relation entretenue par un groupe et sa fraction associée. Il s'agit là du concept d' « encerclement ».

Avant d'aller plus loin, nous rappelons ce qu'est la distance de Hausdorff. Soit A et B deux ensembles, la distance de Hausdorff entre A et B, notée ici h(A, B) est définie de la manière suivante : h(A, B) = max(f(A,B), f(B,A)) avec f(A,B) = max d(a, B) pour a élément de A et d(x, Y) = min d(x, y) pour y élément de Y.

Soit r le type de « séparateurs », r vaut '>' ou '*'. On dit qu'un groupe est « encerclé » d'ordre (n, r) lorsque la *distance de Hausdorff* entre le groupe et sa fraction de type r est égale à n.

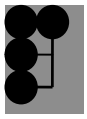

Figure 22

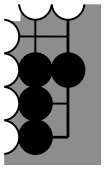

Figure 23

Par exemple, le groupe noir en bas à droite de la figure 1 (figure 22) est « encerclé » d'ordre (1, '>') par la fraction blanche (figure 23). Nous allons montrer maintenant deux exemples plus compliqués, plus représentatifs de ce qu'est l'encerclement.

Figure 24

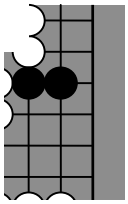

Figure 25

Figure 26

Premier exemple : le groupe noir du bord droit de la figure 1 (figure 24) est encerclé d'ordre (4, '>') par la fraction blanche de la figure 25 et il est encerclé d'ordre (1, '*') par la fraction blanche de la figure 26.

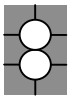

Figure 27

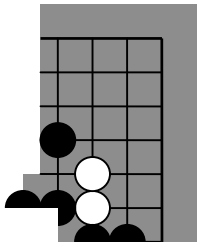

Figure 28

Figure 29

Deuxième exemple : le groupe blanc du coin supérieur droit de la figure 1 (figure 27) est encerclé d'ordre (6, '>') par la fraction noire de la figure 28[1]. De plus, il est encerclé d'ordre (2, '*') par la fraction noire de la figure 29. Ces deux derniers exemples montrent que l' « encerclement » ne se traduit pas par un seul nombre mais par un ensemble de nombres représentant chacun la distance d'encerclement d'un groupe par une fraction d'un type donné. La notion d' « encerclement » est utilisée ensuite pour évaluer l'état d'un groupe. Un programme capable de reconnaître des encerclements d'ordre d'autant plus grand sera d'autant plus fort. La modélisation et l'utilisation de l'encerclement par la fonction d'évaluation reste un sujet de recherche largement ouvert où beaucoup de travaux restent à accomplir. L'infériorité des programmes dans ce domaine précis par rapport aux performances de l'être humain est un des facteurs de faiblesse des programmes de go. Après avoir vu une description topologique[2] d'une position, on peut maintenant décrire la position d'un point de vue morphologique.

4.5. « intérieur », « extérieur » et « groupe-morphologique »

Les joueurs de go utilisent des concepts de « groupe », de « territoire » et d'« influence » . Dans cette partie, nous allons voir comment les modéliser. Pour ce faire nous allons d'abord définir un « groupe-morphologique », son « intérieur » et son « extérieur ». Ces deux dernières notions sont proches des notions intuitives habituelles d'intérieur ou d'extérieur d'un objet. Pour les modéliser, nous allons utiliser des *opérateurs ad hoc* construits par dérivations des opérateurs bien connus de morphologie mathématique [Serra 1982].

Pour la suite, rappelons les différents opérateurs morphologiques que nous utilisons. Soit A un ensemble, on note D(A) et on appelle *dilatation morphologique* de A l'ensemble composé de A et de ses voisins au sens de la 4-connexité. On note E(A) et on appelle *érosion morphologique* de A

[1] Car l'intersection du coin, en haut à droite de la zone, est à une distance 6 du groupe blanc de 2 pierres

[2] On peut objecter à juste titre que la notion d'encerclement n'est pas topologique uniquement car elle fait intervenir la notion métrique de distance.

l'ensemble composé de A moins les éléments de A voisins du complémentaire de A au sens de la 4-connexité. On note Fe(A) et on appelle *frontière externe* de A l'ensemble D(A)-A. On note Fi(A) et on appelle *frontière interne* de A l'ensemble A-E(A). Soit A un ensemble, on appelle *fermeture* morphologique de A et on note F(A) l'ensemble E(D(A)). On appelle *ouverture* morphologique de A et on note O(A) l'ensemble D(E(A)).

Ces définitions étant données, il faut noter une première différence entre la morphologie mathématique et la topologie discrète utilisée par le RS. [Egenhofer & Sharma 1993] et [Galton 1999] définissent l'intérieur d'un ensemble comme étant l'érosion de cet ensemble. Cette définition ne permet pas de définir une topologie au sens mathématique mais seulement une « quasi-topologie » [Galton 1999] et oblige à de nombreux aménagements [Egenhofer & Sharma 1993]. Par exemple, l'intérieur de l'intérieur d'un ensemble n'est pas égal à l'intérieur de l'ensemble. Par opposition, l'ouverture morphologique et la fermeture morphologique respectent la relation d'idempotence, fondamentale lorsque ces opérateurs sont utilisés dans des algorithmes itératifs. C'est une des raisons de l'utilisation de la morphologie mathématique dans notre travail et surtout du succès de celle-ci en traitement d'images.

Pour le go, nous adaptons ces opérateurs à nos besoins de la manière suivante. Premièrement, nous *numérisons* les opérateurs E et D. (Sans numérisation, nous ne sommes pas parvenus à simuler la reconnaissance des territoires des joueurs humains et avec la numérisation nous avons réussi à nous en approcher). Deuxièmement, nous faisons intervenir les *deux couleurs* (Noir et Blanc). Une intersection occupée par une pierre noire (respectivement blanche) est initialisée avec la valeur +128 (respectivement -128). Pour chaque intersection du goban, si une intersection est positive (resp. négative) ou nulle et n'est pas voisine d'une intersection négative (resp. positive), l'opérateur D additionne (resp. soustrait) le nombre de voisines positives (resp. négative) à l'intersection. Pour chaque intersection du goban, si une intersection est positive (resp. négative), l'opérateur E soustrait (resp. additionne) le nombre de voisines négatives (resp. positives) ou nulles à l'intersection en vérifiant qu'elle reste positive (resp. négative) ou nulle.

Puis nous utilisons des opérateurs $X = E*E...*E*D*D...*D$ et $Y = D*D...*D$ où E est composé un certain nombre de fois 'e', et D 'd' fois. Avec $e = d*(d-1) + 1$. Les détails sont donnés dans [Bouzy 1995b]. Donnons un exemple d'exécution de l'algorithme sur un exemple simple. Si la position initiale correspond à la figure 30.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	128	0	0	128	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 30

Alors après trois dilatations, on aura les valeurs de la figure 31.

0	0	0	1	0	0	1	0	0	0
0	0	2	2	2	2	2	2	0	0
0	2	4	6	5	5	6	4	2	0
1	2	6	136	7	7	136	6	2	1
0	2	4	6	5	5	6	4	2	0
0	0	2	2	2	2	2	2	0	0
0	0	0	1	0	0	1	0	0	0

Figure 31

Et après sept érosions supplémentaires, on aura les valeurs de la figure 32.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	128	6	6	128	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 32

Il faut donc remarquer l'apparition de nombres strictement positifs sur deux intersections situées entre les intersections occupées par des pierres noires. Ces deux intersections appartiennent à X(N) et pas à N.

Nous avons choisi 3 dilatations et 7 érosions pour cet exemple précis. Mais d'autres valeurs de 'e' et 'd' sont possibles pourvu qu'elles respectent la relation *e = d*(d-1) + 1*. Dans le programme Indigo, nous avons 4 dilatations et 13 érosions. Par ailleurs, notre algorithme est connu dans le domaine du logiciel libre sous le nom de l'algorithme « 5-21 » car le programme GnuGo (http://www.gnu.org/software/gnugo/devel.html) a réutilisé notre algorithme avec 5 dilatations et 21 érosions. Dans la suite de l'article les exemples seront pris avec les valeurs 4 et 13. La valeur initiale de +-128 affectées aux pierres noires ou blanches est totalement empirique, elle a été choisie afin de résister à un nombre suffisant d'érosions successives[1].

Pour simplifier la communication, dans la suite de ce document nous allons faire l'abus de langage suivant : nous allons appeler fermeture morphologique d'un ensemble A, l'ensemble X(A) et dilatation morphologique l'ensemble Y(A). Soit N (resp. B) l'ensemble des intersections du damier occupées par des pierres noires (resp. blanches). Les « groupes-morphologiques » sont les composantes connexes de X(N) et de X(B) au sens de la 4-connexité. On appelle « squelette » d'un « groupe-morphologique » G et on note S(G), l'ensemble G intersecté avec l'ensemble des pierres de la couleur de G. On appelle « intérieur » d'un « groupe-morphologique » G et on note In(G), l'ensemble G - S(G). Enfin, on appelle « extérieur » d'un « groupe-morphologique » G et on note Ex(G), l'ensemble Y(S(G)) – G.

On remarquera que notre définition d' « intérieur » ne correspond ni à celle de la topologie discrète utilisée en RS [Egenhofer & Sharma 1993] [Galton 99] ni à celle de l'ouverture en morphologie mathématique [Serra 1982]. Dans notre travail, nous avons In(G) = G - S(G). Nous avons choisi la terminologie d' « intérieur » car elle se rapporte à un groupe-morphologique, constitué d'un « squelette » protégeant l'espace « intérieur » du groupe, et pas à un ensemble quelconque d'intersections du damier.

Sur l'exemple précédent, les deux pierres noires appartiennent au même « groupe-morphologique ». Les deux intersections de valeur 6 sont les intersections de l' « intérieur » de ce « groupe-morphologique » et les deux intersections de valeur 128 sont les intersections du « squelette ».

Toutes ces opérations sont effectuées et aboutissent à la réalisation d'une *carte morphologique* de la position. La figure 33 représente la carte morphologique de la position de la figure 1. Les carrés gris foncés (resp. clairs) représentent les « intérieurs » des groupes morphologiques noirs (resp. blancs). Les « squelettes » sont les pierres. Les « extérieurs » des groupes morphologiques, qui rendraient la figure confuse, ne sont volontairement pas représentés.

[1] 128 permet d'appliquer l'algorithme avec 6 dilatations et 31 érosions car 128/4 = 32.

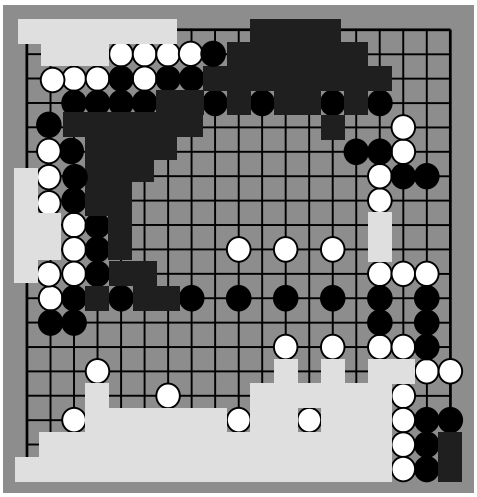

Figure 33

La notion de « groupe » utilisée par les joueurs humains correspond ni à l'une ni à l'autre de nos définitions de « groupe-connecté » ou de « groupe-morphologique » mais un peu à chacune des deux[1]. Concernant les programmes, la question de savoir laquelle des deux notions, « groupe-connecté » ou « groupe-morphologique » est la meilleure est un problème ouvert et surtout une matière de goût du programmeur ; c'est un problème d'intégration dans le reste du programme. On peut préférer la notion de « groupe-connecté » car elle correspond à une notion prouvable par une recherche arborescente. Les problèmes de cette approche sont le temps de réponse à l'exécution, l'extensibilité sans fin de la base des « connecteurs », la mise en avant de la notion de connexion dans le programme, ce qui donne un programme qui connecte ses groupes avant tout. On peut aussi préférer la notion de « groupe-morphologique » car elle est plus intuitive et elle se calcule vite. Par contre, elle n'est adéquate que dans des positions terminales. Cette approche est bonne si la puissance de la machine permet d'atteindre ces positions. Pour notre programme, nous avons une définition de « groupe » qui est la réunion des deux définitions : deux « groupe-connectés » disjoints appartenant à un même « groupe-morphologique » appartiennent au même « groupe » et deux « groupe-morphologiques » disjoints appartenant à un même « groupe-connecté » appartiennent au même « groupe ».

Enfin, pour terminer cette partie, nous faisons correspondre la notion de « territoire », utilisée par les joueurs de go, à la notion d'« intérieur » présentée ici, et celle d'« influence» à celle d' « extérieur ». Dans cette partie, nous avons présenté trois concepts chez les joueurs de go : « groupe », « territoire » et « influence » ; nous avons présentés des outils topologiques et morphologiques qui permettent de définir des approximations de ces trois concepts : « groupe-morphologique », « intérieur » et « extérieur » ; Pour obtenir une évaluation de la position, il manque maintenant la description des concepts qualitatifs.

4.6. « interaction »

Un « groupe » possède des voisins amis et ennemis. Pour chaque couple de groupes ennemis, il est possible de définir une « interaction » qui exprime l'éventuelle domination d'un « groupe » sur l'autre. L'évaluation de la valeur de l'« interaction » entre deux groupes ennemis est obtenue par un raisonnement qualitatif. Elle se traduit par une relation de domination entre deux groupes ennemis. Mais nous choisissons de ne pas alourdir l'exposé de ces relations, très spécifiques du go, et dont la nature n'est pas spatiale.

4.7. « mort », « inversion », « agrégation »

[1] Il nous est impossible de donner cette définition « humaine » de groupes, sinon nous l'aurions fait. C'est en partie ici que réside la difficulté de la programmation du jeu de go : la définition des groupes.

Par définition, une « mort » se produit lorsqu'un « groupe » réunit toutes les conditions suivantes : il ne possède pas un « intérieur » de taille assez grande[1], il ne possède pas de voisin ami, il est dominé par tous ses voisins ennemis, et il est « encerclé ». Dans ce cas, une « inversion » se produit : le groupe « mort » change de couleur et fusionne avec ses groupes voisins. Sur l'exemple de la figure 1, le groupe de la figure 22 est « mort ». La figure 34 montre la description conceptuelle après l' « inversion » conséquence de la mort de ce groupe.

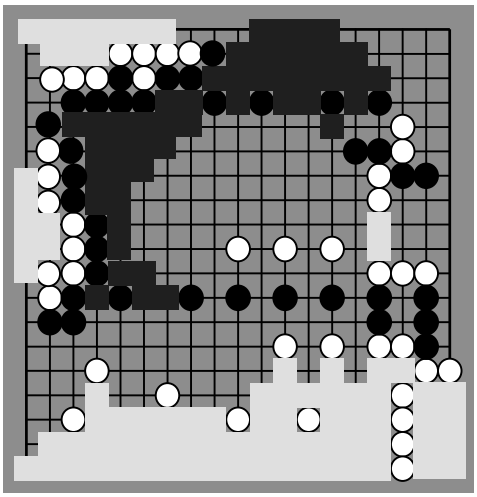

Figure 34

Après une inversion, une agrégation est faite pour fusionner le groupe mort avec les groupes adverses. Ce qui donne un nouveau groupe plus gros.

4.8. Retour sur la fonction d'évaluation

Après avoir listé les concepts spatiaux utilisé par la fonction d'évaluation, il est temps de donner la forme de la FE. C'est très simple : la FE est une boucle sur la construction des « groupes » :

> Tant qu'il existe un groupe « mort », effectuer l' « inversion ».
> Sommer les valeurs des intersections contrôlées (+1 pour Noir et –1 pour Blanc).

Si, au cours d'une boucle, une mort est détectée, le programme « inverse » le groupe mort et agrège les groupes adverses autour de lui pour donner un nouveau groupe. Si aucune mort n'est détectée, la boucle s'arrête et on effectue une somme des contributions de chaque intersection du damier. Une intersection est contrôlée par une couleur lorsqu'elle est occupée par un groupe qui n'est pas mort à la fin de l'exécution de la boucle. Une intersection contrôlée par Noir (respectivement par Blanc) apporte une contribution de 1 point (respectivement –1 point) à la FE qui est donc égale à la somme des contributions de chaque intersection du goban.

La richesse des concepts spatiaux présentés dans cette partie et leur résumé en une unique valeur de FE est troublant. Mais il faut savoir que les concepts spatiaux décrits ici servent non seulement à l'élaboration de la FE mais aussi à la GC qui n'est pas l'objectif de cet article.

Actuellement, les programmes exécutent la construction de la FE sur 19x19 en un dixième de seconde environ sur les PC actuels (Pentium 600 Mhz avec 64 Mo). Cette performance dépend évidemment du contenu de la FE, variable d'un programme à l'autre, et des positions testées. Dans certains programmes, cette performance est également due à l'optimisation de la FE avec l'incrémentalité.

---

[1] Cette condition est approximative et, à nouveau, nous choisissons de ne pas alourdir la présentation l'article de concepts liés au go et non à l'espace. Pour une condition exacte, se référer à [Chen & Chen 1999].

4.9. « incrémentalité »

Incrémentalité signifie ne pas remettre à jour la FE de façon complète à chaque coup mais seulement la partie de la FE qui a changé. Pour des raisons de rapidité, ce principe est fondamental dans la programmation du go. En effet, un coup joué en un endroit donné ne produit des changements qu'au voisinage du coup mais en dehors de celui-ci, le coup ne produit pas d'effet : la description conceptuelle ne change pas. Après un coup joué sur une position dont on connaît déjà la description conceptuelle totale, il est alors utile que la machine ne remette à jour que la partie de la description qui change. Pour cela, on définit l' « empreinte » d'un ensemble d'intersections comme étant l'ensemble sur lequel un coup joué produira un changement de l'état de cet ensemble d'intersections. Ainsi, lorsque un coup est joué sur une intersection, la machine détruit les ensembles dont l'empreinte rencontre le coup, puis reconstruit la nouvelle description. Ce mécanisme permet de gagner un facteur 2 sur 9x9, 4 sur 13x13 et 10 sur 19x19, ce qui n'est pas négligeable [Bouzy 1997]. Mais le problème de la spécification des empreintes pour chaque type d'objets reste difficile, voire problèmatique. En fait, on retrouve par là, un symptôme du « frame problem » bien connu en IA [Lenting & Brasppening, 1993] [Pylyshyn 1987].

5. DISCUSSION

Dans cette partie, nous donnons d'abord une crédibilité expérimentale au contenu de cet article en donnant les résultats du programme de go que nous avons développé et qui a servi de point d'ancrage pour écrire ce document. Ensuite, puisque l'état de l'art en RS est constitué pour une large part de RS qualitatif, nous posons la question de savoir si la PdG est un exemple de RS plutôt qualitatif ou plutôt quantitatif et nous tentons d'y répondre. Enfin, la partie précédente ayant montré l'apport de la morphologie mathématique et de la connexité à la PdG, il est normal de répondre à la question de savoir ce que la PdG apporte en retour à l'ensemble du RS.

5.1. Validation expérimentale

Pour apporter un élément de validation au contenu de ce document, nous pensons qu'il est nécessaire de dire que celui-ci se base sur l'expérimentation d'un programme de go de niveau international : le programme Indigo. En effet, une première version (Indigo93) avait été développée au cours d'une thèse sur la modélisation cognitive du joueur de go [Bouzy 1995]. Depuis, d'autres versions ont été réalisées (http://www.math-info.univ-paris5.fr/~bouzy/INDIGO.html) et correspondent de très près à la description donnée ici. Indigo est classé sur l'échelle internationale des programmes de go (au 7 septembre 2001, il est classé 8ème sur 16 sur l'échelle 19x19 et 7ème sur 16 sur l'échelle 9x9) situé sur Internet (http://www.cgl.ucsf.edu/go/ladder.html). De plus, il a terminé 10ème sur 17 à la coupe Ing 1998 à Londres et 13ème sur 16 à la coupe Ing 1999 à Shanghai (http://www.britgo.demon.co.uk/ing/), championnat du monde des programmes de go. Il a également terminé 5ème sur 6 au dernières olympiades des jeux de réflexion dans la compétition des programmes de go (http://www.britgo.org/results/mso2000/computer.html). Nous estimons son niveau à 15ème kyu environ. La fonction d'évaluation de Indigo utilise de nombreux concepts spatiaux décrits dans cet article. Nous pensons que la description de la fonction d'évaluation donnée dans cet article est représentative de celles des autres programmes de go.

5.2. Qualitatif ou quantitatif ?

Le RS s'applique en général à des domaines complexes et continus où il est important de savoir si celui-ci est quantitatif ou qualitatif. Bien sûr, en remarquant simplement que le damier est de taille limitée, on peut objecter que le raisonnement au go est purement qualitatif. Cependant, il nous semble très intéressant de cerner quels sont les endroits du programme où celui-ci fait un raisonnement d'une nature se rapprochant d'un type de raisonnement ou de l'autre. Donc par précaution, pour répondre à la question de savoir si le RS de la PdG est qualitatif ou quantitatif, nous dirons, dans ce paragraphe, qu'un concept est qualitatif si sa valeur est dans un ensemble de cardinal inférieur à trois tel que par exemple {vrai, faux} ou {perdu, indéterminé, gagné} et qu'un concept est quantitatif si il se traduit par un nombre entier. Par exemple le score d'une partie est un nombre, donc le score est un concept quantitatif.

Un « groupe-connecté » est un concept qualitatif, utilisé dans le RS *qualitatif* d'un programme de go. Nous allons étudier la relation « fait-partie-du-meme-groupe-connecté » et montrer comment est utilisée cette relation dans la construction des groupes de la FE. Cette relation s'applique aux « groupes-connectés » en cours d'agrégation de la FE (cf paragraphe 4.2). C'est-à-dire que si, à un instant donné de la construction de la fonction d'évaluation, on a deux « groupes-connectés » g1 et g2 en cours de construction et vérifiant g1 « fait-partie-du-meme-groupe-connecté » g2 car le programme a identifié une relation d'appartenance '>' à un même groupe connecté entre ces deux groupes, alors le programme crée un « groupe-connecté » g3 = g1 ∪ g2. Le problème est que, faisant cela, le programme n'est pas toujours correct car, au go, cette relation « fait-partie-du-même-groupe-connecté » *n'est pas transitive*.

Nous allons en donner les raisons. Si g1 « fait-partie-du-même-groupe-connecté » g2 est vraie et si g2 « fait-partie-du-même-groupe-connecté » g3 est aussi vraie, on peut trouver des exemples où g1 « fait-partie-du-même-groupe-connecté » g3 est faux. En d'autres termes, si g1 ∪ g2 est considéré comme connexe et si g2 ∪ g3 est considéré comme connexe, alors, aussi contre-intuitif que cela puisse paraître, g1 ∪ g2 ∪ g3 n'est pas toujours connexe ! L'exemple le plus simple est celui du connecteur '>' de la figure 35, appelé « kosumi » dans le jargon du go :

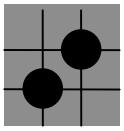

Figure 35

Il est constitué de deux groupes noirs, composés chacun d'une pierre noire. C'est un connecteur '>' car si Blanc joue sur une intersection, Noir joue sur l'autre et devient connecté explicitement au sens de la règle du jeu. Les deux groupes noirs sont donc reliés par la relation « fait-partie-du-même-groupe-connecté ». La figure 36 est alors intéressante à étudier.

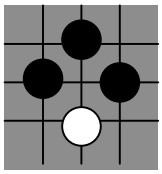

Figure 36

Si l'on suppose transitive la relation « fait-partie-du-même-groupe-connecté », on peut penser qu'il y a un seul groupe noir. En effet, la pierre noire de gauche de la figure 36 est reliée à celle du haut par la relation « kosumi » de la figure 35. La pierre noire de gauche et la pierre noire du haut sont donc reliées par la relation « fait-partie-du-même-groupe-connecté ». Ces deux pierres noires forment donc un ensemble connexe dans notre modèle de connexion '>'. De même, la pierre noire de droite de la figure 36 est reliée à celle du haut par la relation « kosumi » de la figure 35. La pierre noire de droite et la pierre noire du haut sont donc reliées par la relation « fait-partie-du-même-groupe-connecté ». Ces deux pierres noires forment donc aussi un ensemble connexe dans notre modèle de connexion '>'. Si l'on suppose transitive la relation « fait-partie-du-même-groupe-connecté », alors les trois pierres noires de la figure 36 forment un ensemble connexe pour notre modèle de connexion '>'.

La réalité du go est autre. En effet, si Noir joue le premier, il se connecte explicitement, mais si Blanc joue le premier, il peut déconnecter noir en deux groupes distincts comme le montre la séquence de la figure 37.

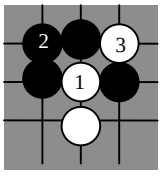

Figure 37

Ce contre-exemple, dans un domaine bien plus simple que le monde réel, montre, selon nous, une difficulté à mettre en pratique le RSQ. Evidemment, les problèmes de connexion existent également en RS [Cohn & Varzi 1999].

L' « encerclement » est une relation entre un « groupe » et une « fraction », qui sont des concepts qualitatifs. Mais l'encerclement possède des ordres qui sont des nombres (quantitatif) : un « groupe » est encerclé à une distance donnée. L' « encerclement » est une relation *semi-qualitative* (elle utilise des concepts qualitatifs) *et semi-quantitative* (son résultat est un ensemble de nombres). Les concepts morphologiques « groupe-morphologique », « intérieur » et « extérieur », calculés numériquement, sont évidemment *quantitatifs*.

Pour déterminer les propriétés d'un « groupe », des questions quantitatives sont posées : combien de composantes connexes l'intérieur du groupe possède-t-il ? combien d'amis le groupe possède-t-il ? L'utilisation répétée du mot « combien » signifie la présence de quantités. A cet endroit, le raisonnement est donc *quantitatif*. Les valeurs des propriétés des groupes permettent alors de calculer une valeur qualitative ('<', 'indéterminé','>') pour déterminer si un groupe est « mort ». Le concept de « mort », met donc en lumière un RS *qualitatif*.

Enfin, la somme des valeurs attribuées à chaque intersection (+1 ou –1) est calculée. La FE, est finalement au niveau ultime, un exemple de raisonnement *quantitatif*.

Donc, en observant l'alternance d'inférences qualitatives et quantitatives, on peut dire que le calcul de la FE au go n'est ni un RS qualitatif ni un RS quantitatif pur, mais plutôt un raisonnement hybride.

La construction d'une description qualitative d'une position est nécessaire à la PdG. En effet sans elle, les programmes seraient perdus dans la combinatoire du jeu et ne pourraient pas choisir de coup « intéressant » du tout. Evidemment, le fait que la description qualitative, obtenue en pratique par les programmes, soit une approximation de descriptions théoriques parfaites, entraîne des erreurs des programmes effectuant un raisonnement qualitatif sur cette description (cf. exemple de la non transitivité de la connexion ci-dessus). Mais ici, n'incriminons pas le RS qualitatif car quelle autre technique ferait mieux que lui ? En bout de chaîne, le RS qualitatif ne suffit pas puisque le but du jeu, et donc la fonction d'évaluation, sont purement quantitatifs : il s'agit de contrôler un ensemble d'intersections *plus grand* que celui de l'adversaire.

6. CONCLUSION

Après avoir brièvement présenté l'état de l'art du RS, nous avons montré que le blocage du niveau des programmes de go à un niveau moyen sur l'échelle humaine n'est pas seulement dû à la complexité combinatoire mais aussi à la difficulté de construire un formalisme adéquat et complet pour calculer les caractéristiques spatiales de la fonction d'évaluation. Nous avons ensuite décrit les nombreux concepts spatiaux, que fait intervenir une fonction d'évaluation au go ; les principaux sont le regroupement, le fractionnement, l'encerclement, l'agrégation. Nous avons montré que le go est une excellente illustration des théories du RS en raison de cette richesse conceptuelle. Nous avons montré, pour chaque concept, quels sont les outils mathématiques utilisés pour les simuler sur machine (morphologie mathématique, topologie, distance de Hausdorff, RSQ). Enfin, nous pensons que la démarche expérimentale de la PdG, basée sur une validation informatique des modèles construits sur un domaine réglé, évaluable et plus simple que le monde réel, est complémentaire des approches théoriques de modélisation formelle de l'espace réel.

7. REFERENCES

[Allen 1983], James Allen, *« Maintaining knowledge about temporal intervals »*, Communications of the ACM, vol. 26 (11), november 1983, pp. 832-843.

[Allis 1994] Louis Victor Allis, *« Searching for solutions in Games and Artificial Intelligence »*, Ph.D. thesis, Vrije Universitat Amsterdam, Maastricht, September 1994, http://www.cs.vu/~victor/thesis.html

[Asher & Vieu 1995], Laure Vieu et Nicolas Asher, *« Toward a geometry of common sense : a semantics and a complete axiomatization of mereotopology »*, Proceedings IJCAI-95, Montréal, 1995.

[Benson 1976], D.B. Benson, *« Life in the game of go »*, Information Sciences, 10, pp.17-29, 1976.

[Berlekamp 1991], Elwin Berlekamp, *« Introductory overview of Mathematical go Endgames »*, Proceedings of symposia in applied mathematics, 43, 1991.

[Berlekamp & Wolfe 1994], Elwin Berlekamp, David Wolfe, *« Mathematical go Endgames, Nightmares for the Professional go Player »*, Ishi Press International, San José, London, Tokyo, 1994.

[Boon 1989], Mark Boon, *« Pattern Matcher for goliath »*, Computer go 13, winter 89-90.

[Boon 1991], Mark Boon, *« Overzicht van de ontwikkeling van een go spelend programma »*, Afstudeer scriptie informatica onder begeleiding van prof. Bergstra J, Amsterdam, 1991.

[Bouzy 1995a], Bruno Bouzy, *« Modélisation cognitive du joueur de go »*, thèse de l'université Paris 6, 13 janvier 1995, http://www.math-info.univ-paris5.fr/~bouzy

[Bouzy 1995b], Bruno Bouzy, *« Les Ensembles Flous au jeu de go »*, Actes des Rencontres francophones sur la Logique Floue et ses Applications, LFA95, pp 334-340, Paris 1995.

[Bouzy 1997], Bruno Bouzy, *« Incremental updating of objects in Indigo »*, Proceedings of the Fourth Game Programming Workshop in Japan, pp. 179-188, Hakone, 1997.

[Buro 1994], M. Buro, *«* Methods *for the evaluation of game positions using examples »*, Ph.D. thesis of the University of Paderborn, Germany, 1994.

[Cazenave 1996] Tristan Cazenave, *« Système d'apprentissage par auto-observation. Application au jeu de go »*, thèse de l'université Paris 6, 13 décembre 1996.

[Chen & Chen 1999] Ken Chen, Zixingh Chen, *« Static analysis of life and death in the game of go »*, Information Sciences, 121, (1-2), pp. 113-134, 1999.

[Chen 2000] Ken Chen, *« Some practical techniques for global search in go »*, ICGA Journal, 23 (2) , pp.67-74, 2000.

[Clarke 1981], B. Clarke, *« A calculus of individuals based on connection »*, Notre Dame Journal of Formal Logic, 22, 3, pp. 204-218, 1981.

[Clementini & Di Felice 1995], Eliseo Clementini, Paulino Di Felice, *« A comparison of methods for representing topological relationships »*, Information Sciences 3, pp. 149-178, 1995.

[Cohn 1996] Anthony Cohn, *« Calculi for spatial resoning »*. In : J. Pfalzgraf, J. Calmet, J.A. Campbell, editor, Artificial Intelligence and Symbolic Mathematical Computation, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 1138, pp. 124-143, 1996.

[Cohn 1997] Anthony Cohn, *« Qualitative spatial representation and reasoning techniques »*. Proceedings KI'97, Berlin: Springer-Verlag, Lecture Notes in Artificial Intelligence No. 1303, pp. 1-30, 1997.

[Cohn & Varzi 1999] Anthony Cohn, A.C. Varzi, *« Modes of Connection »*, COSIT'99, In Freksa and Mark (eds), Cognitive and Computational Foundations of Geographic Information Science, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 1661, pp. 299-314, 1999.

[Conway 1976], John Conway, *« On Number And Games »*, Academic Press, 1976.

[Conway & al. 1982], John Conway, Elwin Berlekamp, Richard Guy, *« Winning ways »*, Tome 1 et 2, Academic Press, 1982.

[Egenhofer 1989] Max Egenhofer, *« A formal definition of binary topological relationships »*. in : W. Litwin and H.J. Schek, editors, Third FODO conference, Paris, France, Springer-Verlag, Lecture Notes in Computer Science No. 367, pp. 457-472, 1989.

[Egenhofer 1991] Max Egenhofer, *« Reasoning about binary topological relations »*. Proceedings SSD'91, Advance in Spatial Databases, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 525, pp. 143-160, 1991.

[Egenhofer & Sharma 1993] Max Egenhofer, Jayant Sharma, *« Topological Relations Between Regions in R^2 and Z^2»*. 3rd ISLSD, in Abel and Ooi (eds), Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 692, pp. 316-336, 1993.

[Egenhofer & Mark 1995] Max Egenhofer, David Mark, *« Naive geography »*. COSIT'95: Conference on Spatial Information Theory, Semmering, Austria. In Frank, A. U. and Kuhn, W., editors, Spatial Information Theory: A Theoretical Basis for GIS, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 988, pp. 1-16, 1995.

[Enzenberger 1996] Markus Enzenberger, *«The Integration of A Priori Knowledge into a go Playing Neural Network* », http://www.cip.physik.uni-muenchen.de/~enz/Neurogo/Neurogo.html, Décembre 1996.

[Erwig & Schneider 1997] Martin Erwig, Markus Schneider, *« Partition and Conquer »*, COSIT'97, In Hirtle and Frank (eds), Spatial Information Theory: A Theoretical Basis for GIS, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 1329, pp. 389-407, 1997.

[Fotland 1992], David Fotland, *« Many Faces of Go, documentation and playing algorithm »*, 1992.
[Fotland 1996], David Fotland, *« Computer Go Design Issues »*, Message envoyé le 1er octobre 1996 à computer-go@anu.edu.au.
[Fraenkel & Lichtenstein 1981], Fraenkel A.S., Lichtenstein S., *« Computing a perfect strategy for n by n Chess requires time exponential »*, Journal of Combinatorial Theory, Serie A, Vol. 31, N°2, September 1981, pp. 199-214.
[Galton 1999] Antony Galton, *« The Mereotopology of Discrete Space »*, COSIT'99, In Freksa and Mark (eds), Cognitive and Computational Foundations of Geographic Information Science, Berlin: Springer-Verlag, Lecture Notes in Computer Science No. 1661, pp. 250-266, 1999.
[Hernandez 1994] Daniel Hernandez, *« Qualitative Representation of Spatial Knowledge »*, Berlin: Springer-Verlag, Lecture Notes in Artificial Intelligence No. 804, 1994.
[Herskovits 1982], A. Herskovits, *« Space and the preposition in english : regularities and irregularities in a complex domain »*, Ph. D., Stanford University, 1982.
[Hsu & al. 90], FH Hsu, TS Anantharaman, Murray Campbell, Andreas Nowatzyk, *« A grandmaster chess machine »*, Scientific American, Vol 263, N 4, October 1990.
[Kuipers 1978], Benjamin Kuipers, *« Modelling spatial knowledge »*, Cognitive Science, 2, 129-153, 1978.
[Kuipers & Byun 1988], Benjamin Kuipers, Yung-Tai Byun, *« A robust qualitative method for robot spatial learning »*, Proceedings of the AAAI-88, pp. 774-779, 1988.
[Kraszek 1988], Janus Kraszek, *« Heuristics in the life and death algorithm »*, Computer go 9, Winter 88-89.
[Lenting & Brasppening 1993], Lenting J., Brasppening P., *« The Frame Problem from an Engineering Perspective »*, Int. Journal of Software Engineering and Knowledge Engineering 3, 2, pp. 257-285, 1993.
[Lesniewski 1927], S. Lesniewski, *« O podstawach matematyki »*, Przeglad Filosoficzny, vols. 30-34, 1927-1931.
[Mano 1984], Y. Mano, *« An approach to conquer difficulties in developping a go playing program »*, Journal of Information Processing, 7, 2, 1984.
[Marr 1982], David Marr, Vision, Freeman, 1982.
[Mueller 1993], Martin Mueller, *« Game theories and computer go »*, Computer go workshop, Cannes, 1993.
[Mueller 1995], Martin Mueller, *« Computer go as a sum of local games »*, PhD thesis, ETH Zurich, fevrier 1995, http://nobi.eth.ch/~martin/martin.html
[Mueller 1998] Martin Mueller, *« Computer go: A Research Agenda »*, Proceedings CG-98 (eds H.J. van den Herik and H. Iida), Lecture Notes in Computer Science N 1558, Springer Verlag, Heidelberg, Germany, ISBN 3-540-65766-5, pp. 252-264, 1998.
[Mueller 1999] Martin Mueller, *« Decomposition Search: A Combinatorial Games Approach to Game Tree Search, with Applications to Solving go Endgames »*, in: Proceedings IJCAI'99, 1, pp. 578-583, 1999.
[Mueller 2000a] Martin Mueller, *« Not like other games - why tree search in go is different »*, in: Proceedings of the 5th Joint Conference on Information Sciences, 2000.
[Mueller 2000b] Martin Mueller, *« Review: Computer go 1984-2000 »*, in: Proceedings of the 2th International Conference on Computer and Games, Hamamatsu, Japan, pp. 421-430, 2000.
[Mukerjee & Joe 1990], Amithaba Mukerjee & Gene Joe, *« A qualitative model for space »*, Proceedings of AAAI-90, MIT Press, Cambridge (MA), pp . 721-727.
[Pylyshyn 1987], Pylyshyn Z.W., *« The Robot's Dilemma : the Frame Problem in Artificial Intelligence »*, Ablex Publishing Corporation, 1987.
[Randell & al. 1992], D. Randell, Z. Cui, A. Cohn, *« A spatial logic based on regions and connection »*, Proceedings of KR-92, Morgan Kaufmann, San Matéo, 1992.
[Robson 1982], Robson J.M., *« The Complexity of go »*, TR-CS-82-14, Australian National Unviversity Department of Computer Science, 1982.
[Serra 1982], Jean Serra, *« Image analysis and mathematical morphology »*, Academic Press, London, 1982.
[Shirayanagi 1989], Shirayanagi, *« Knowledge representation and its refinement »*, Computer go, n°11, summer 1989.
[Thomsen 2000] Thomas Thomsen, *« Lambda-search in game trees – with application to go »*, in: Proceedings of the 2th International Conference on Computer and Games, Hamamatsu, Japan, pp. 57-79, October 2000.
[Thorp & Walden 1972], E.O. Thorp, W.E. Walden, *« A computer-assisted study of go on MxN boards »*, Information Sciences, 4, pp. 1-33, 1972.
[Wilcox 1979], Bruce Wilcox, *« The structure of the go program Interim.2 »*, Proceedings of the IJCAI-79, 1979.
[Wolf 1994], Thomas Wolf, *« The program goTools and its computer-generated tsume go database »*, Proceedings of the First Game Programming Workshop in Japan, pp. 84-96, Hakone, 1994.
[Wolf 2000] Thomas Wolf, *« Forward pruning and other heuristic search techniques in tsume go »*, Information Sciences, 122, pp. 59-76, 2000.
[Zobrist 1969], Albert Zobrist, *« A model of visual organisation for the game of go»*, Proceedings AFIPS 34, pp. 103-112, 1969.

8. ANNEXE. REGLES DU JEU

Cette annexe contient un résumé minimal des règles du jeu de go, qui doit suffire pour comprendre l'essentiel des connaissances sur le go contenues dans cet article. Le lecteur souhaitant avoir des règles complètes, mais peut-être moins claires, pourra se référer aux nombreux livres qui les décrivent.

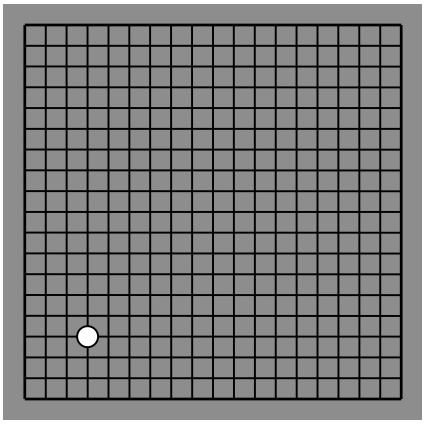

Figure A1

Une partie se joue à deux avec des « pierres » noires et blanches que l'on pose sur le damier, appelé aussi « goban ». A son tour, un joueur pose une pierre une « intersection » comme le montre la figure A1. Il existe une règle de capture qui permet d'enlever des pierres du damier. La figure A2 montre quatre exemples de positions locales illustrant cette règle.

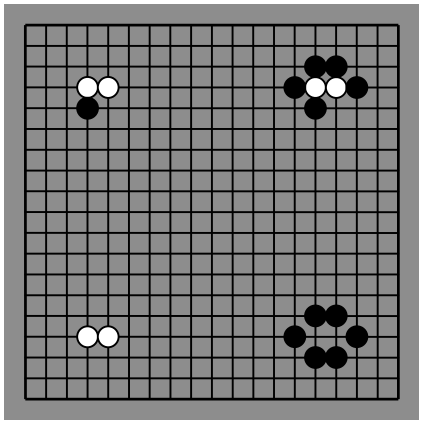

Figure A2

Tout d'abord, une « chaîne » de pierres est un ensemble connexe de pierres de la même couleur. Par exemple, en bas à gauche de la figure A2 se trouve une chaîne blanche constituée de deux pierres blanches. Ensuite, une chaîne possède des « libertés » qui sont les intersections voisines et vides de la chaîne. Par exemple, la chaîne blanche en bas à gauche de la figure A2 possède six libertés et la chaîne blanche en haut à gauche de la figure A2 en possède cinq. En haut à droite, elle n'en possède plus qu'une seule. On dit qu'elle est « atari ». Enfin, lorsqu'un joueur supprime la dernière liberté d'une chaîne en jouant dessus, la chaîne est « capturée » et enlevée du goban. Par exemple, en bas à droite du de la figure A2, une chaîne blanche de deux pierres a été capturée par Noir.

La partie se déroule par alternance de coups noirs et blancs. Les situations répétitives sont interdites. Cela signifie qu'un coup joué ne doit pas amener à une position antérieure. Ce qui est rendu possible par la règle de capture. Grâce à la règle de capture, un joueur est capable de contrôler des intersections. La partie s'arrête d'un commun accord lorsque les deux joueurs contrôlent tout le damier. Le joueur gagnant est celui qui contrôle plus d'intersections que l'adversaire. Sur la position de la figure A3 toutes les intersections sont contrôlées.

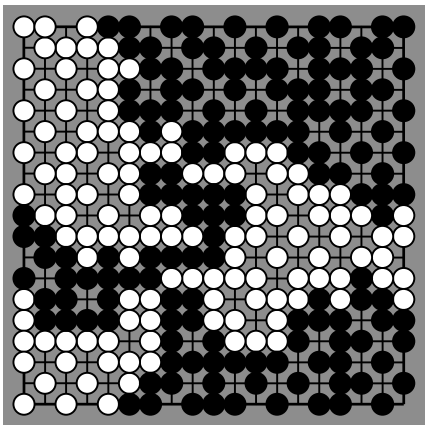

Figure A3

A titre indicatif, on peut compter les points. Le groupe blanc en haut à gauche contrôle 68 intersections. Celui situé en bas à gauche en contrôle 31 et celui situé au milieu a droite 64. Blanc contrôle donc au total 163 intersections. Le groupe noir situé en haut à droite contrôle 99 intersections, celui situé au milieu à gauche en contrôle 37 et celui situé en bas à droite 62. Le nombre d'intersections contrôlées par Noir vaut donc 198. Finalement, Noir gagne de 35 points.

Sur la position de la figure A3, il faut remarquer que le contrôle des intersections est *explicite* : le contrôle d'une intersection par une couleur se traduit soit par la présence d'une pierre de cette couleur soit par l'impossibilité de poser une pierre de l'autre couleur dessus. Sur cette position explicite, les joueurs ne peuvent donc qu'être d'accord et la partie s'arrête. Il faut aussi noter que cette position est atteinte au bout d'un nombre de coups très grand et que les deux joueurs peuvent être d'accord sur le contrôle total du damier bien avant d'aboutir à cette position. La figure A4 montre un exemple de damier où la partie s'arrête beaucoup plus tôt que sur la position précédente parce que les joueurs sont d'accord sur le contrôle du damier. Sur cette position, le contrôle des intersections est *implicite*. En effet, le joueur d'une couleur suppose que si le joueur de l'autre couleur venait contester ses intersections, il serait capable de capturer les chaînes adverses.

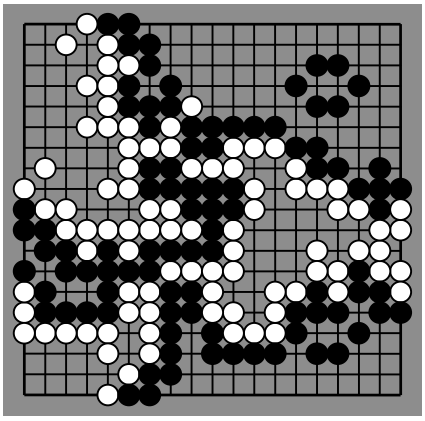

Figure A4

Si les deux joueurs n'étaient pas d'accord, ils continueraient de jouer sur les intersections conflictuelles jusqu'à prouver explicitement leur contrôle. C'est l'expérience qui permet de reconnaître de mieux en mieux le contrôle des intersections et d'arrêter la partie de plus en plus tôt. Alors que la définition du contrôle explicite des intersections est simple, il est beaucoup plus difficile de définir le contrôle des intersections correspondant à la position de la figure A4.