Un modèle du jeu de Go basé sur des interactions

Bruno Bouzy

LAFORIA-IBP
Université Paris VI Tour 46 2ème étage
4, place Jussieu 75252 PARIS Cedex 05
Tél.: (+33) 1 44 27 70 10 Fax: (+33) 1 44 27 70 00
e-mail : bouzy@laforia.ibp.fr

Abstract:

This paper addresses the interaction-based model of the game of Go on which the INDIGO program relies. The agents of the model are groups of intersections. They are built with connexity and morphology. One one hand, they interact with the external world, then they inherit an interior and an exterior. The (agent-world) interactions allow to find the approximate weaknesses of agents. One the other hand, the agents interact between each other. The (agent-agent) interactions allow to find the state (dead or alive) of the agents that were already recognized as weak. When a group is dead, the predators cluster around the dead group and eat it to create a new agent. The global state of a position is built with the local states of local agents. This interaction-based model gives to INDIGO a correct behaviour in actual games against other Go playing programs that join the international Computer Go Ladder.

Keywords :

Computer Go, agent, interaction, spatial autonomy.

Résumé

Cet article présente le modèle du jeu de Go basé sur des interactions qui supporte le programme INDIGO. Les agents sont les groupes d'intersections d'une position. Ils sont construits par connexité et morphologie. D'une part, ils interagissent avec le monde externe et, partant de là, ils héritent d'un intérieur et d'un extérieur. Les interactions (agent - monde externe) permettent de reconnaître approximativement les faiblesses des agents. D'autre part, les agents interagissent les uns avec les autres. Les interactions inter-agents permettent de connaître l'état (mort ou vivant) des agents déjà reconnus comme approximativement faibles. Les agents prédateurs d'un agent mort fusionnent autour de lui et le mangent pour former un nouvel agent. L'état global d'une position est déduit de l'état local de chacun des agents. Ce modèle d'interaction donne à INDIGO un comportement très honorable dans les parties réelles contre d'autres programmes participant à l'échelle internationale des programmes de Go.

Mot-clés :
Programmation du jeu de Go, agents, interaction, autonomie spatiale.

1. Introduction

Le but de cet article est de décrire le modèle du jeu de Go, basé sur des interactions, qui supporte le programme de Go INDIGO [Bouzy 1995]. Il s'agit de comparer, sur l'exemple du jeu de Go, le résultat d'un système où les agents interagissent seulement avec le monde, avec celui d'un système où les agents interagissent avec le monde mais aussi entre eux.

Le plan de l'article comporte donc deux parties principales. La première partie (§3.) est ontologique : elle permet de définir ce que ceux *sont* les agents au sein d'un "monde". Elle montre comment sont construits les agents (§3.1. et §3.2.), comment ils se différencient du monde par distinction d'un intérieur et d'un extérieur (§3.3.), comment ils sont qualifiés à partir de leurs interactions agent-monde seules (§3.4.). Cette partie montre donc comment un monde est occupé par une société d'agents. L'état global de la société d'agents est construit au moyen des états des agents individuels. La figure 1 symbolise un monde (en gris) où des agents (en blanc ou en noir) se différencient du monde. Les interactions agent-monde sont bien présentes au travers de la frontière des agents (le trait noir qui sépare le blanc ou noir du gris).

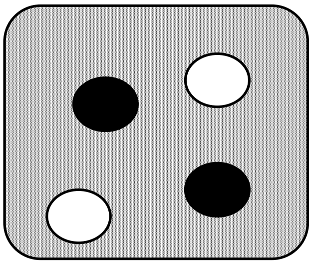


figure 1

La deuxième partie principale (§4.) est phylologique : elle décrit les interactions entre les agents de la société (§4.2.) après avoir décrit les limites de l'approche ontologique seule (§4.1.). Les interactions permettent de qualifier les agents précisément dans les nombreuses positions où le modèle sans interaction inter-agents échoue (§4.3.). Par conséquent, elle montre comment une position au Go est interprétée globalement à partir de ce modèle sur l'interaction inter-agents. La figure 2 symbolise le monde de la figure précédente enrichi des interactions inter-agents (les doubles flèches noires).

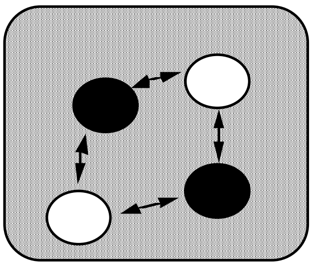


figure 2

Avant ces deux parties principales et à la lumière des résultats des applications développées avec des systèmes multi-agents dans des domaines autres que le jeu de Go, nous citons d'autres modélisations multi-agents envisageables au jeu de Go (§2.) mais laissées de côté par cet article. Avant de conclure, nous donnons les résultats actuels du programme INDIGO et nous discutons de la nature des agents (§5.).

Pour préserver l'unité de cet article, nous avons choisi de ne pas couper son enchaînement par des notions intermédiaires. Cependant, le lecteur soucieux d’une compréhension complète, pourra lire les annexes présentant les règles du jeu de Go (§7.), la notion de "jeu" (§8.) ou la morphologie mathématique (§9.).

2. Préliminaire

Le jeu de Go[1] est un jeu de somme nulle à deux personnes à information complète. Le nombre moyen de coups possibles sur une position 19x19 est environ 200 et la durée d'une partie est d'environ 300 coups. Une position est difficile à évaluer et nécessite beaucoup de connaissances. La complexité du jeu de Go (10^{172}) est nettement supérieure à celle des Echecs (10^{43}). Ce qui interdit l'utilisation de la force brute comme aux Echecs. Le meilleur programme de Go, GOLIATH, n'a que le niveau d'un joueur très moyen. Sur ce constat, toutes les approches pour programmer le Go sont les bienvenues, en particulier l'approche multi-agents comme nous allons le voir. Ce paragraphe cite deux façons d'utiliser un modèle multi-agents sur le jeu de Go, soit en modélisant l'être humain (§2.1.), soit en modélisant le monde (§2.2.). Il rappelle les grandes directions utilisées par différentes applications récentes utilisant des systèmes multi-agents.

2.1. Modéliser un joueur ou expert humain

Dans notre thèse [Bouzy 1995], nous avons élaboré un modèle cognitif du joueur de Go où l'agent privilégié était le joueur lui-même. Dans ce paragraphe, nous montrons deux façons de modéliser un joueur de Go humain.

2.1.1. Un système bi-joueurs

Si l'on pose :

Un agent = un joueur

on modélise des agents humains qui communiquent. Dans le traitement du langage naturel, [Brassac 1994] est un exemple où un agent est un être humain qui communique. Il serait intéressant d'étudier la communication qui existe entre deux joueurs de Go mais cela ne sera pas traité dans cet article.

2.1.2. Un système multi-experts

L'idée est d'utiliser la propriété du jeu de Go d'être multi-points de vue. Des experts spécialisés dans des points de vue différents et interagissant peuvent interpréter une position de Go mieux qu'ils ne le feraient chacun tout seul. L'idée [Pompidor 1994] est donc de poser :

Un agent = un expert

Les experts sont typés. Un expert est spécialiste de la reconnaissance des formes, un autre expert est spécialiste de la recherche arborescente, un autre expert est spécialiste global. Les experts communiquent entre eux pour choisir le coup à jouer. Cette idée multi-experts se répand en IAD étant donné la complexité du monde réel et les multiples points de vue que doivent posséder les applications. Dans [Boissier & al. 1994], des agents typés utilisent un langage d'interaction pour comprendre une scène.

2.2. Modéliser l'espace, un système multi-groupes

L'idée que nous suivons dans cet article est de modéliser l'espace (plutôt que de modéliser le joueur). Les applications des systèmes multi-agents où les agents sont spatiaux commencent à être développées. Le logiciel SIGMA de généralisation cartographique [Baeijs & al. 1995] en

[1]Les règles du jeu figurent en annexe.

est un bon exemple. Dans INDIGO, les agents sont spatiaux. Ils sont situés sur le goban. *Chaque agent sera un groupe d'intersections*. Dans la suite de cet article, nous posons :

Un agent = un groupe[1].

3. Ontologie d'une position de Go

Cette partie définit un agent situé (§3.1.), puis un agent connexe (§3.2.) et enfin un agent "morphologique" (§3.3.). Avec les deux outils de connexion et de morphologie, des agents se distinguent du monde par apparition d'un intérieur et d'un extérieur. Elle montre enfin les apports de ce modèle basé sur l'interaction agent-monde seule sans interaction inter-agents (§3.4.).

Nous intercalons les déclarations des classes C++ utilisées par INDIGO lorsque les concepts correspondant sont présentés.

La figure 3 est un exemple de position de Go sur 19x19 avec des pierres noires et blanches posées sur les intersections du goban.

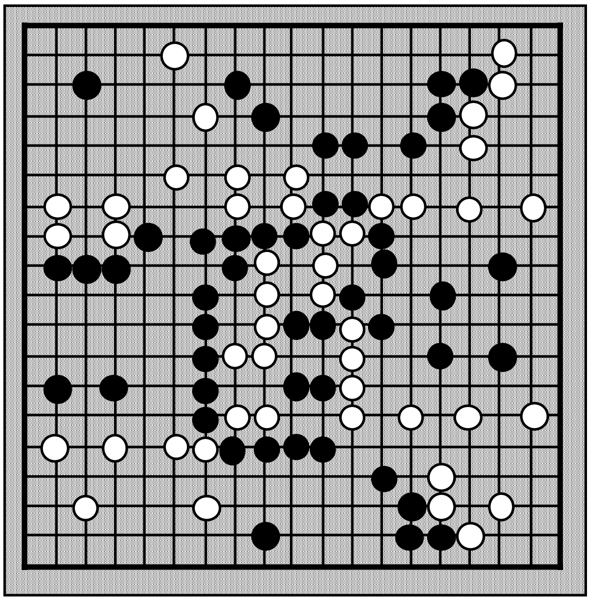


figure 3

Le joueur de Go reconnaît des groupes de pierres, base de son raisonnement. Ces groupes de pierres sont les agents de notre modèle.

3.1. Un agent situé

La propriété de base d'un agent est d'être situé sur un ensemble d'intersections.

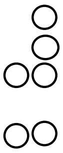
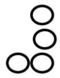
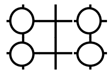
[1]Défini dans les parties suivantes.

```
class AgentSitue : public Agent
{
        EI1 * situation;
};
```

L’attribut ‘situation’ de la classe ‘AgentSitué’ contient l’ensemble des intersections sur lesquelles repose l’agent.

3.2. Un agent connexe

Le modèle supportant INDIGO considère les connexions '>'[2] comme des éléments qui permettent de fusionner les groupes entre eux.

Par exemple, le groupe:  est construit par la fusion des chaînes  et  par l’intermédiaire de la connexion '>' .

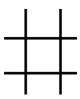
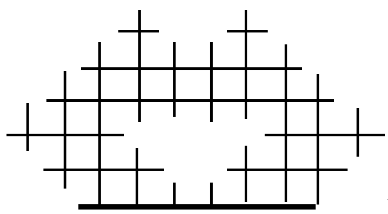
La connexion est fondamentale au Go car elle permet de construire les groupes sur lesquels le raisonnement ultérieur est basé.

```
class AgentConnexe : public AgentSitue;
```

Cette définition de la connexion est un moyen de résoudre le problème de la fusion des données au jeu de Go. La fusion de données est un domaine de recherche utilisé en robotique ou en systèmes d'information spatiaux. Un cadre multi-agents englobant un système de fusion de données appliqué à la robotique est décrit dans [Ayari & Haton 1995].

3.3. Un agent morphologique[3]

Le modèle supportant INDIGO considère l'ensemble des intersections appartenant à la fermeture morphologique d'un agent connexe mais n'appartenant pas à l'agent connexe, comme étant l'intérieur d'un nouveau type d'agent : l'agent morphologique. Il considère l'ensemble des intersections appartenant à la dilatation morphologique d'un agent connexe mais n'appartenant pas à l'agent connexe, comme étant l'extérieur de l'agent morphologique.

Le groupe  possède  pour intérieur et  pour extérieur[4].

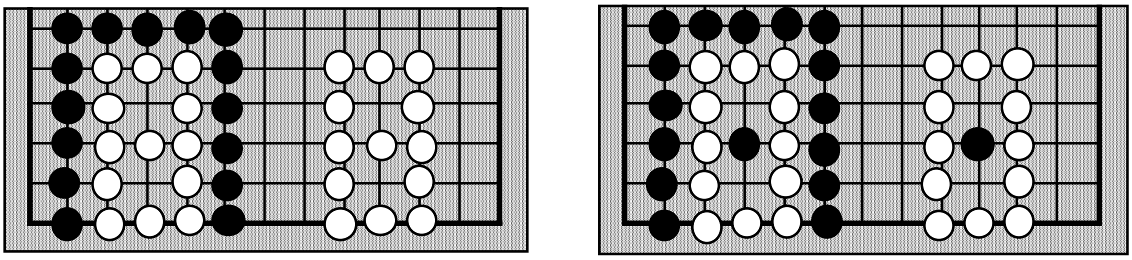
[1]EI est un type défini dans INDIGO qui signifie Ensemble d’Intersections.
[2]cf. annexe sur la notion de jeu.
[3]cf. annexe sur la morphologie mathématique.
[4]Idem.

```
class AgentMorphologique : public AgentConnexe
{
        EI * interieur;
        EI * exterieur;
        int etat-interieur;
        int etat-exterieur;
};
```

3.3. Les apports de l'approche ontologique seule

L'approche précédente permet de définir spatialement où se situent les agents au sein du monde. Certaines intersections appartiennent à l'intérieur d'un agent morphologique d'autres restent extérieures aux agents. En étudiant qualitativement et quantitativement l'intérieur et l'extérieur d'un agent, il est possible de déterminer si son état est dépendant des futurs changements dans son voisinage ou bien s'il en est indépendant, c'est-à-dire si l'agent est *autonome spatialement*.

Les agents peuvent être autonomes spatialement ou non. Sur la figure 4, les groupes blancs A et B sont autonomes spatialement et les groupes blancs C et D non autonomes spatialement.



A B C D

figure 4

En effet, les deux libertés intérieures des groupes blancs A et B sont inaccessibles à Noir. Que le groupe blanc soit totalement encerclé ou pas du tout n'a pas d'importance. La vie du groupe blanc est indépendante de son extérieur, ce que nous résumons en disant que ces deux groupes blancs sont autonomes spatialement.

Par contre, les deux libertés intérieures des groupes blancs C et D sont accessibles à Noir. Que le groupe blanc soit encerclé ou non a une importance extrême. La vie du groupe blanc dépend de son extérieur, ce que nous résumons en disant que ces deux groupes blancs ne sont pas autonomes spatialement.

Au Go, il existe des lois ou proverbes grossiers qui permettent de connaître la viabilité d'un groupe en fonction de l'état de l'intérieur du groupe et de l'état de l'extérieur du groupe. Sur la figure 4, les groupes blancs A et C ont un extérieur dont l'état est nul, les groupes blancs B et D ont un extérieur dont l'état est suffisamment grand, les groupes blancs A et B ont un intérieur dont l'état est égal à deux, les groupes blancs C et D ont un intérieur dont l'état est égal à un. Nous pouvons représenter ces groupes sur un graphique avec deux axes : un axe vertical représentant l'état de l'intérieur du groupe et un axe horizontal représentant l'état de l'extérieur du groupe. La figure 5 montre la position des groupes A, B, C et D de la figure 4 et l'existence de 3 zones : l'une où les groupes sont autonomes spatialement, une autre où les groupes sont stables par absence de voisins, et une autre proche de l'origine O où les groupes sont faibles ou morts. Le point origine O correspond à la capture de pierres au sens de la règle du jeu.

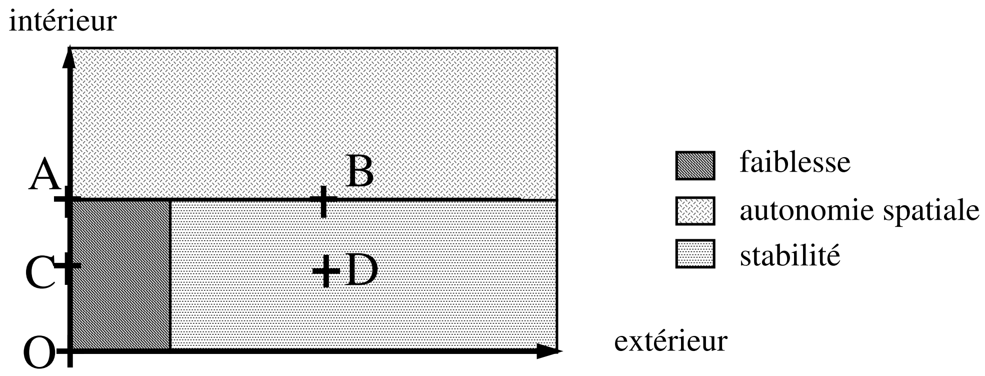


figure 5

Le programme RISIKO [Wolf 1992] résout des problèmes de vie et de mort. Ses résultats sont remarquables puisqu'il résout ces problèmes avec des performances égales à celles de très bons joueurs amateurs. Il a également trouvé quelques erreurs dans les recueils de problèmes de vie et mort édités par les écoles de joueurs professionnels. RISIKO utilise de la recherche arborescente car la complexité des problèmes traités correspond à la puissance des machines actuelles.

Par contre, RISIKO ne résout des problèmes que lorsque les groupes sont complètement encerclés par des groupes adverses eux-mêmes complètement vivants. Sur le graphique de la figure 5, ces problèmes correspondent au segment [AO].

4. Phylologie d'une position au Go

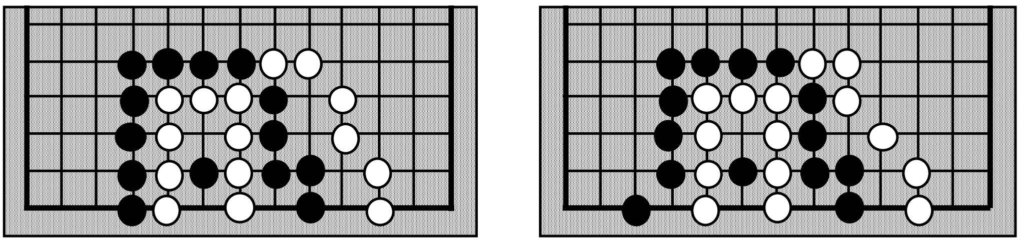
Cette partie définit un ensemble d'agents voisins interagissant les uns avec les autres. Après avoir montré les limites de l'approche précédente (§4.1.), elle décrit les deux types d'interaction inter-agents possibles au Go (§4.2.) : l'*interaction amie* et l'*interaction ennemie*. Sur une exemple, elle montre comment l'état d'un agent est déterminé (§4.3.).

4.1. Limites de la seule approche ontologique :

Le rectangle de la figure 5 où les groupes sont faibles est le principal obstacle pour la programmation du jeu de Go car :

- en tout point du rectangle différent de l'origine, il existe des exemples de groupes vivants et des exemples de groupes morts,
- la complexité des problèmes situés dans ce rectangle est trop grande pour utiliser la force brute.

Les positions de la figure 6 montrent quatre groupes faibles appartenant au rectangle de la figure 5. Les groupes F et G sont vivants et les groupes E et H sont morts. Les groupes K, L, M et N sont stables car ils possèdent un état extérieur suffisamment grand.



K E F L M G H N

figure 6

Notre programme INDIGO s'attaque à ces problèmes de vie et mort au sens large[1]. Il résout une partie de ces problèmes en faisant intervenir son modèle d'interaction multi-agents. Une partie seulement car INDIGO, limité par la puissance des machines actuelles, n'utilise pas de recherche arborescente. Ces problèmes sont fréquents en parties réelles. Le fait de trouver une solution non optimale sur ces problèmes est un avantage important face à des programmes qui recherchent en vain une solution optimale et laissent finalement le problème de côté.

Sur la figure 6, les agents E, F, G et H possèdent tous un état intérieur ‘<‘ et un état extérieur ‘<‘. En effet, en un coup, il ne peuvent posséder deux libertés intérieures inaccessibles, ni sortir de l'encerclement. Les agents K, L, M et N possèdent tous un état extérieur ‘>‘. La figure 7 positionne ces groupes suivant la représentation de la partie précédente. Les groupes K, L, M et N possèdent un état intérieur flou car celui-ci dépend de l'échelle à laquelle il est reconnu. Ce qui explique le trait vertical sur la figure 7 pour les représenter.

[1]Ils font intervenir l'état de l'extérieur d'un groupe en plus de son état intérieur.

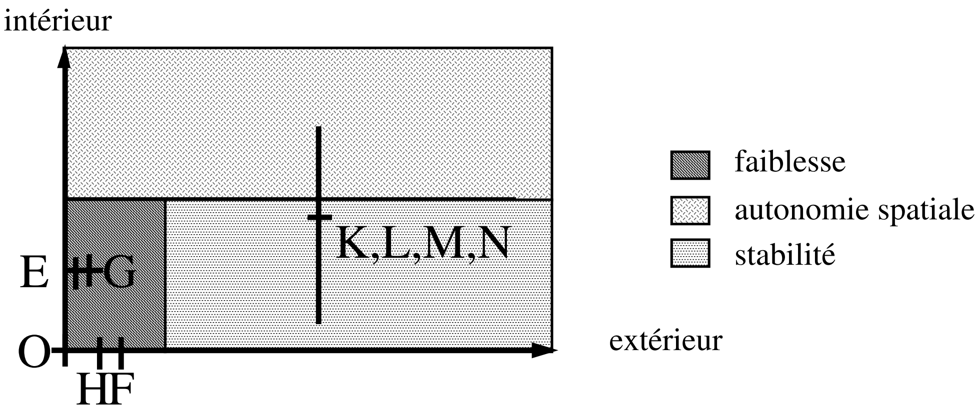


figure 7

4.2. Les interactions possibles

Étant donné qu'un agent est soit Noir, soit Blanc, les interactions possibles sont les interactions amies et les interactions ennemies. En C++, dans INDIGO l'interaction est définie comme suit :

```
class Interaction
{
        Agent * agent1;
        Agent * agent2;
        int etat;
};


class AgentPhylologique : public AgentMorphologique
{
        Liste[1] * interactions-amies;
        int etat-amical;
        Liste * interactions-ennemies;
        int etat-inimical;
        int viabilité;
};
```

4.2.1. Une interaction amie

Deux agents voisins qui peuvent se connecter en un coup et former un agent unique ont une interaction amie '*'. Chaque agent possède donc un état amical A qui synthétise l'état des différentes interactions amies de l'agent.

Sur l'exemple de la figure 6, aucun agent ne possède d'ami. Chaque agent de la figure 6 possède donc une amitié '<'.

4.2.2. Une interaction ennemie

A ce stade, il est possible de définir un triplet T(agent) = f(I, E, A) pour chaque agent où I est son état intérieur, E son état extérieur, A son état amical.

[1]Liste est un type défini dans INDIGO pour implémenter les listes.

Il existe une relation d'ordre partiel entre les deux triplets de deux agents adverses.

Par exemple, si un agent 1 possède un triplet (I_1, E_1, A_1) et un agent 2 possède un triplet (I_2, E_2, A_2) avec $I_1 < I_2$, $E_1 < E_2$, $A_1 < A_2$, alors l'interaction ennemie de l'agent1 avec l'agent 2 est '<'.

Dans beaucoup de cas, il est difficile de comparer deux triplets. Par exemple, lorsque $I_1 = I_2$ = '<', $A_1 = A_2$ = '<' et E_1 presque égal à E_2 non nul[1]. Nous avons utilisé le symbole pour exprimer que la comparaison est difficile.

Dans beaucoup d'autres cas comme ceux de l'exemple de la figure 6, il paraît impossible de comparer les triplets deux à deux car nous avons :

$I_E = I_F = I_G = I_H$ = '<',
$E_E = E_F = E_G = E_H$ = '<',
$A_E = A_F = A_G = A_H$ = '<'.

Cependant en nombre de libertés nous avons :

$I_E = 2$, $I_F = 0$, $I_G = 2$, $I_H = 0$,
$E_E = 1$, $E_F = 5$, $E_G = 2$, $E_H = 4$.

INDIGO utilise ces différences pour dire que :

interaction(E,F) = '<',
interaction(H,G) = '<'.

Les règles qui permettent d'aboutir à cette conclusion figurent dans [Bouzy 1995].

D'autre part, puisque $E_K = E_L = E_M = E_N$ = '>', nous avons de façon triviale :

interaction(E,K) = '<',
interaction(F,L) = '<',
interaction(G,M) = '<',
interaction(H,N) = '<'.

4.3. La synthèse des interactions, la mort.

Le nombre d'interactions amies ou ennemies n'est pas limité. Chaque agent possède un état d'inimitié qui synthétise l'état des différentes interactions ennemies de l'agent. La règle fondamentale qui détermine la mort d'un agent est la suivante :

Si pour tout voisin B d'un agent A, T(B) > T(A) alors état(A) = '<' et l'agent A est mort.

Sur l'exemple de la figure 6, les agents E et H sont morts car il n'ont que des interactions ennemies '<'. Les agents F et G sont vivants par conséquence. La figure 8 symbolise cet état.

---

[1]L'encerclement d'un groupe est un concept intuitif difficile à reconnaître finement.

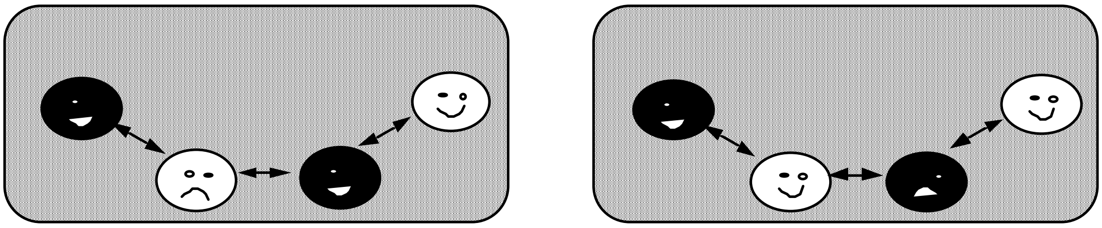


figure 8

Pour résumer : chaque agent mesure sa viabilité en fonction de l'état des interactions avec ses voisins.

Les agents prédateurs voisins de l'agent mort fusionnent en un seul et nouvel agent. Les intersections sur lesquelles est situé le nouvel agent est la réunion des intersections sur lesquelles sont situés les agents prédateurs. Les ensembles d'intersections 'situation', 'intérieur' et 'extérieur' de l'agent proie créent désormais l'ensemble 'intérieur' du nouvel agent. La figure 9 symbolise cet état.

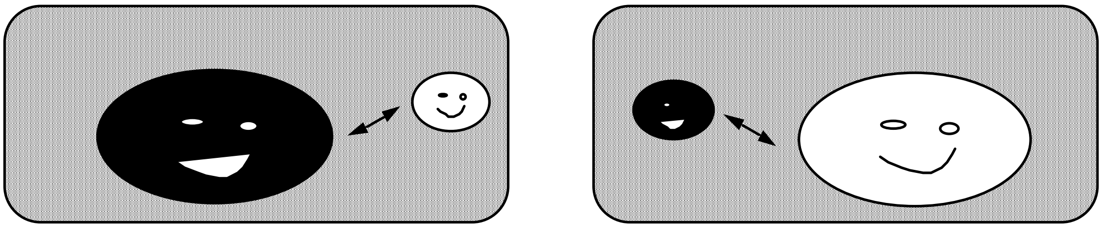


figure 9

L’état global du goban est déterminé par une boucle de détection d’agents morts. A chaque mort d’agent, un nouvel agent est créé (cf. ci-dessus); cela modifie l’ensemble des agents, donc l’ensemble des états des agents, donc la boucle continue. La boucle s’arrête lorsqu’aucune mort d’agent n’est détectée. L’état global résultant est chaotique car des cascades de morts d’agents peuvent survenir. Un exemple figure dans [Bouzy 1995].

5. Résultats pratiques et discussion du modèle

Cette partie précise les résultats pratiques internationaux de INDIGO (§5.1.) et discute de l'intérêt de notre étude pour les systèmes multi-agents (§5.2.).

5.1. Résultats pratiques de INDIGO

INDIGO utilise le modèle présenté dans cet article. Il a été développé en C++. Il comprend 35 000 lignes de C++ et 500 règles. Il tourne sur des stations Sun sparcstation. Il joue un coup entre quelques secondes et une minute. Il est classé honorablement sur la Computer Go Ladder [Petersen 1994] : 4ème sur 7 sur l'échelle 19x19 et 8ème sur 11 sur l'échelle 9x9 au 15 Novembre 1995. Sur l'échelle, les meilleurs programmes sont EGO et MANY FACES OF GO [Fotland 1992]. Le niveau de INDIGO est estimé entre 16ème et 20ème kyu. Les meilleurs programmes du monde sont GOLIATH [Boon 1991], HANDTALK qui est écrit entièrement en assembleur (!?), GO INTELLECT [Chen 1990] et STAR OF POLAND [Kraszek 1988]. RISIKO, le programme qui résoud des problèmes de vie et mort, ne joue pas de partie complète et ne participe pas non plus à cette échelle.

5.2. Discussion

Nous avons décrit des agents dans le monde particulier du jeu de Go. Ce paragraphe résume et discute les propriétés des agents.

1) L'agent est *situé*, ce qui est fondamental. Tout être vivant est situé.

2) L'agent possède un *intérieur* et un *extérieur* qui le déterminent.

3) L'agent est *social* car son état dépend de ses interactions avec ses voisins.

4) L'agent est *gourmand*. Il peut manger ses voisins pour survivre comme dans le règne animal. Il doit ensuite digérer sa proie. A ce titre, il est à la fois proie et prédateur.

5) Est-il *autonome* ? L'agent situé sur des intersections est autonome spatialement dans certains cas. L'agent informatique avec ses 500 règles construites par le programmeur ou bien par d'autres programmes n'est évidemment pas autonome.

Finalement, les agents décrits dans cet article sont *situés*, *sociaux*, *gourmands et autonomes spatialement*, et possèdent un *intérieur* et un *extérieur*.

6. Conclusion

Nous avons présenté le modèle du jeu de Go, basé sur des interactions, qui supporte le programme de Go INDIGO. Nous avons présenté le modèle en deux parties distinctes. D'une part, nous avons montré comment les agents se définissent par opposition avec leur environnement par création d'un *intérieur* et d'un *extérieur* en utilisant des principes de connexité et de morphologie. Nous avons montré comment INDIGO reconnaît la stabilité, l'autonomie spatiale ou la faiblesse d'un agent avec les seules interactions agent - monde externe. Cette première phase est cruciale car l'identification correcte des agents détermine l'adéquation ou non de la seconde phase. Par contre, cette phase ne permet pas de connaître l'état des agents de façon précise. D'autre part, nous avons montré comment les agents interagissent entre eux. Les interactions agent-agent permettent, elles, de connaître précisément l'état d'un agent. INDIGO résoud des problèmes issus de partie réelles en utilisant ce mécanisme d'*interaction* entre les agents. Celui-ci lui procurant une solution non optimale très pratique en parties réelles contre d'autres programmes de Go.

7. La règle du jeu de Go

Dans ce paragraphe, nous présentons le jeu de Go au travers d'un exemple de partie. Ensuite nous donnons les règles du jeu sous forme plus précise.

7.1. Une partie de Go

Une partie se joue à deux joueurs, **Noir** et **Blanc**. Une partie se joue sur un damier appelé **goban** constitué de 19 lignes et 19 colonnes qui se coupent suivant 361 **intersections**. Pour simplifier, nous donnons un exemple de partie jouée sur 9x9 avec 81 intersections seulement. Au début de la partie le goban est **vide**. Noir commence en posant une **pierre** noire sur une intersection, par exemple avec ❶ comme sur la figure *RDJ-1*:

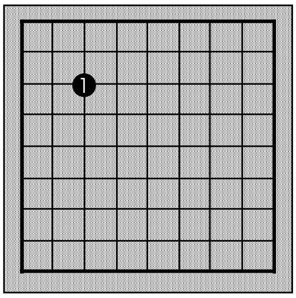


figure RDJ-1

La pierre posée sur le goban est voisine de 4 intersections vides: les **libertés** qui lui permettent de respirer. On ne peut pas jouer de coup sur une intersection où l'on n'a pas de liberté. Une pierre posée sur une intersection du bord (respectivement coin) du goban possède 3 (resp. 2) libertés.

Noir et Blanc posent à tour de rôle des pierres sur le goban. A partir de la position de la figure *RDJ-1*, la partie peut par exemple se poursuivre comme sur la figure *RDJ-2*:

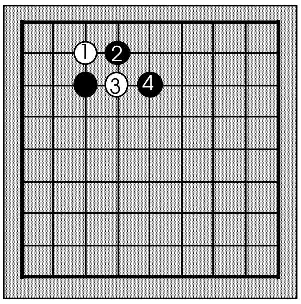


figure RDJ-2

Ici, ❹ a fait **atari** à ③ Cela signifie que ③ ne possède plus qu'une seule liberté. A partir de cette position, deux cas sont possibles. Blanc peut défendre sa pierre avec ① :

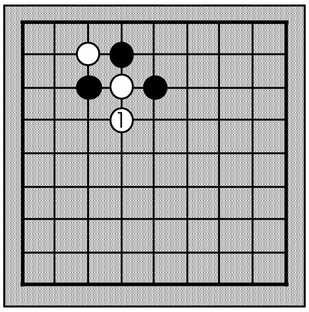


figure RDJ-2a

Sur la figure *RDJ-2a*, Blanc a constitué une **chaîne** de 2 pierres. Le sort des 2 pierres est lié à celui de la chaîne, elle mettent en commun leur libertés. Sur la figure *RDJ-2a*, la chaîne blanche a 3 libertés.

On peut attaquer les pierres adverses en les capturant, c'est-à-dire en supprimant la dernière liberté d'une chaîne de pierres. Par exemple, si Blanc joue ailleurs, Noir peut **capturer** la pierre blanche comme sur la figure *RDJ-2b* :

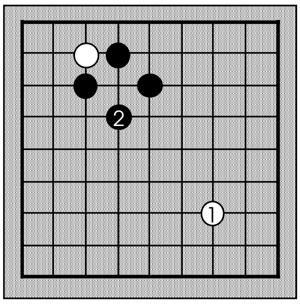


figure RDJ-2b

Les pierres capturées sont enlevées du goban.

La partie se poursuit, le goban se remplit de pierres comme par exemple sur la figure *RDJ-3* :

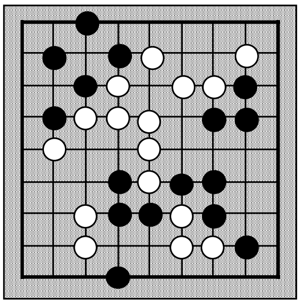


figure RDJ-3

La partie se termine quand Noir et Blanc **passent**. A ce moment là, on compte le **score**. On attribue un **point** par intersection **contrôlée** au joueur qui la contrôle. Une intersection est contrôlée par une couleur si elle est occupée par une pierre de la couleur ou si, étant vide, lorsqu'une pierre de l'autre couleur y sera posée, elle sera capturée. Le joueur qui a le plus de points **gagne** la partie.

Par exemple, sur la figure *RDJ-4*. toutes les intersections appartiennent enfin soit à Noir, soit à Blanc :

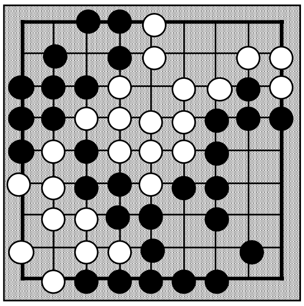


figure RDJ-4

Les deux joueurs passent et comptent les points. La figure *RDJ-5* montre quel joueur contrôle quelles intersections. Les ● et les ■ désignent les intersections contrôlées par Noir et les ○ et les □ désignent des intersections contrôlées par Blanc.

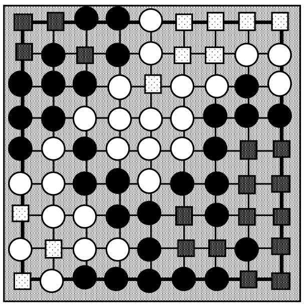


figure RDJ-5

Ici, Noir contrôle 46 intersections et Blanc contrôle 35 intersections. Noir gagne de 11 points.

Au moment où les deux joueurs passent, ils peuvent ne pas être d'accord sur l'appartenance des intersections à l'un et à l'autre. Si c'est le cas, ils continuent de jouer jusqu'à être **d'accord**. Par exemple, si Blanc conteste l'appartenance d'intersections à Noir, il joue dedans comme dans la figure *RDJ-6* :

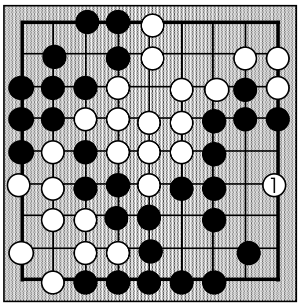


figure RDJ-6

Noir peut capturer la pierre blanche et aussi envahir les territoires blancs. Toutes les intersections peuvent être contestées tant que le contrôle n'est pas suffisamment explicite. La figure *RDJ-8.* montre une position où le contrôle est explicite. La partie s'arrêtera à ce moment car il y a de fortes chances que les joueurs soient d'accord !

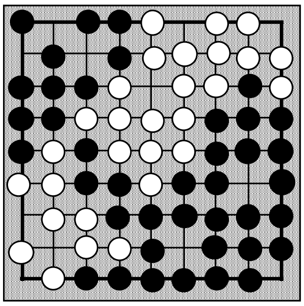


figure RDJ-8

La figure *RDJ-8* montre explicitement l'appartenance des intersections à Blanc et à Noir. Le lecteur vérifiera que Noir gagne toujours de 11 points comme sur la figure *RDJ-5*.

7.2. Des précisions

Le lecteur a senti que les règles du jeu de Go présentées au travers de la partie sur 9-9 sont simples.

Nous pouvons les résumer ainsi :

> (Rdj1) On joue à tour de rôle sur les intersections,
> (Rdj2) On peut capturer l'adversaire en supprimant la dernière liberté d'une chaîne,
> (Rdj3) On peut défendre ses pierres en les connectant sous forme de chaîne,
> (Rdj4) On gagne en contrôlant plus d'espace que l'adversaire.

Nous avons omis volontairement la règle des situations répétitives:

> (Rdj5) On ne peut pas jouer un coup qui donne une position déjà rencontrée dans la partie.

En pratique, la situation du "ko" est la situation répétitive la plus fréquente.

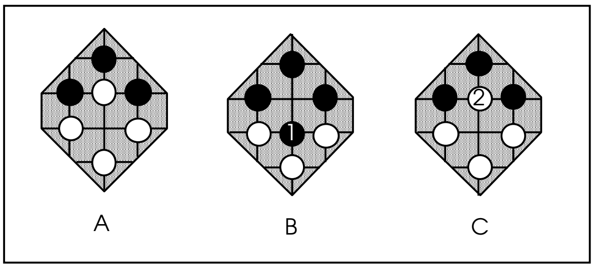


figure RDJ-ko

Dans la position A de la figure *RDJ-ko*, si Noir prend la pierre blanche comme sur la position B, Blanc n'a pas le droit de recapturer au coup suivant : la position C serait une répétition de la position A. Pour jouer en 2 et reprendre la pierre noire 1, Blanc doit d'abord jouer ailleurs, ce qui fait changer l'état du reste du goban. Si Noir joue aussi ailleurs, l'ensemble du goban aura changé et Blanc aura le droit de jouer en 2.

La règle de fin de partie:

> (Rdj6) La partie s'arrête quand les deux joueurs passent et sont d'accord sur le contrôle de l'espace. Si les joueurs ne sont pas d'accord,

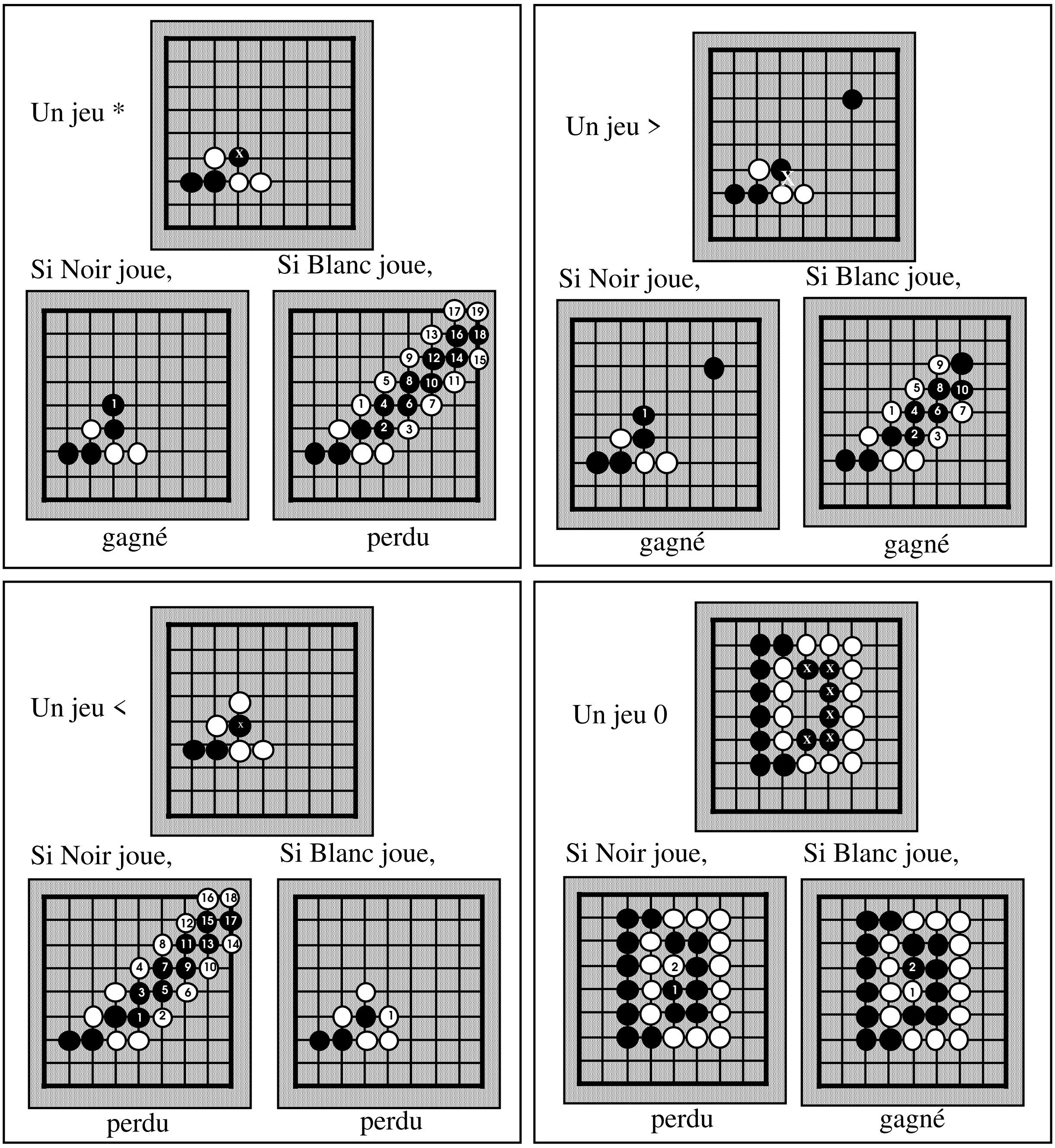
ils continuent de jouer jusqu'à être d'accord.

8. La notion de jeu dans INDIGO

Dans INDIGO, un jeu est associé à une action sur un objet reconnu sur le goban [Bouzy 1995]. Par exemple, le jeu de la chaîne est associé à l'action de capturer ou stabiliser une chaîne. Le joueur de la couleur de la chaîne gagne le jeu de la chaîne si la chaîne obtient 3 libertés ou plus et il perd si la chaîne est physiquement enlevée du goban. Le concept de jeu permet de faire des calculs ou recherches arborescentes associés à une action sur un objet. Nous rappelons ci-dessous un résumé des symboles utilisés pour décrire l'état dynamique d'un jeu.

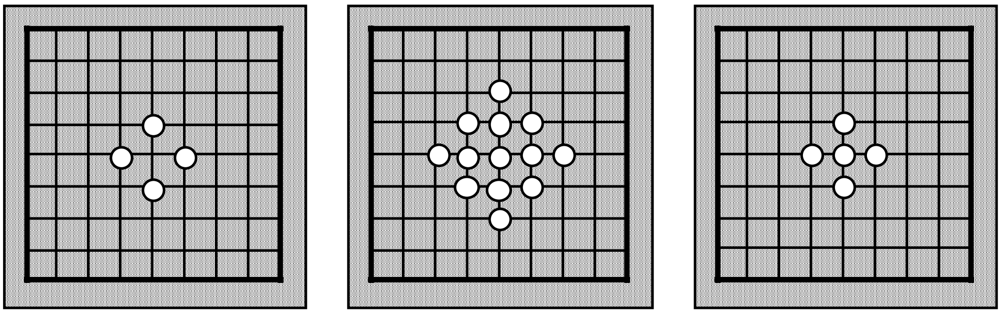
> : le jeu est gagné même si l'adversaire commence,
***** : le premier qui joue gagne,
< : le jeu est perdu même si on commence,
0 : le premier qui joue perd (impasse).

Cette classification est héritée de [Conway 1982].


Un jeu *
Si Noir joue,
Si Blanc joue,
gagné
perdu
Un jeu >
Si Noir joue,
Si Blanc joue,
gagné
gagné
Un jeu <
Si Noir joue,
Si Blanc joue,
perdu
perdu
Un jeu 0
Si Noir joue,
Si Blanc joue,
perdu
gagné

9. Morphologie mathématique

Dans notre thèse, nous avons montré comment la morphologie mathématique permettait de reconnaître les "territoires" au moyen de la fernmeture mophologique et l'"influence" au moyen de la dilatation morphologique. Cette annexe résume la situation avec un minimum d'explications. Le lecteur soucieux du détail pourra trouver ce qu'il cherche dans [Bouzy 1995] ou dans [Bouzy 1995a].



A B C

figure MM-1

Si le damier de gauche est l'ensemble de départ A, celui du milieu est l'ensemble dilaté B=d(A) et celui de droite est l'ensemble fermé C=e(B)=e(d(A))=f(A) (l'ensemble dilaté érodé). B-C repésente l'influence ou l'extérieur et C-A représente le territoire ou l'intérieur. Le principe est simple. Il est affiné avec de la morphologie mathématique floue pour donner des résultats sur des positions réelles où les territoires et influences doivent être reconnus à des échelles variables comme sur la figure MM-2.

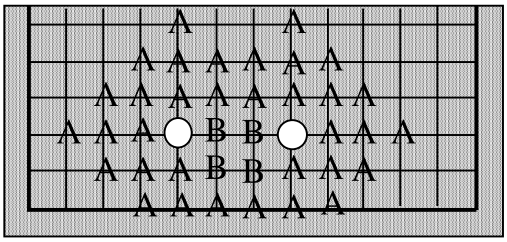


figure MM-2

Sur la figure MM-2, INDIGO applique l'opérateur flou $x(7,3) = e^7 \text{ o } d^3$.

Après trois dilatations :

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

Après sept érosions supplémentaires :

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	4	4	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Ce qui explique très sommairement les A (pour l'influence ou l'extérieur) et les B (pour le territoire ou l'intérieur) de la figure MM-2 et par conséquence la construction de l'intérieur et l'extérieur d'un agent morphologique à une échelle donnée.

10. Bibliographie

[Ayari & Haton 1995] - I. Ayari, J.P. Haton - Cadre pour la fusion multi-capteurs - Troisièmes Journées Francophones IAD&SMA - Chambéry Saint-Badolph - Mars 1995

[Baeijs & al. 1995] - C. Baeijs, Y. Demazeau, L. Alvares - Application des Systèmes Multi-Agents à la Généralisation Cartographique - Troisièmes Journées Francophones IAD&SMA - Chambéry Saint-Badolph - Mars 1995

[Boissier & al. 1994] - O. Boissier, Y. Demazeau, G. Masini, H. Skaf - Une architecture multi-agents pour l'implémenation du bas niveau d'un système de compréhension de scènes - Deuxièmes Journées Francophones IAD&SMA - Voiron - Mai 1994

[Boon 1991] - M. Boon - Overzicht van de ontwikkeling van een Go spelend programma - Afstudeer scriptie informatica onder begeleiding van prof. J. Bergstra - 1991

[Bouzy 1995] - B. Bouzy - Modélisation cognitive du joueur de Go - Thèse de doctorat de l'Université Paris 6 - Janvier 1995

[Brassac 1994] - C. Brassac - L'interaction inter-agents : non littéralité et processualité - Deuxièmes Journées Francophones IAD&SMA - Voiron - Mai 1994

[Chen 1990] - K. Chen - Move decision process of Go intellect - Computer Go 14, spring 90

[Conway 1982] - J. Conway, E.R. Berlekamp, R. Guy - Winnings ways - tome 1 & 2 - Academic press - 1982

[Fotland 1992] - D. Fotland - Many Faces of Go, documentation and playing algorithm - 1992

[Kraszek 1988] - J. Kraszek - Heuristics in the life and death algorithm - Computer Go n°9, winter 88-89

[Petersen 1994] - The Computer Go Ladder - http://cgl.ucsf.edu/go/ladder.html

[Serra 1982] - J. Serra - Image Analysis and Mathematical Morphology - Academic Press - London - 1982

[Wolf 1992] - T. Wolf - Tsumego with Risiko - School of Math. Sciences, Queen mary & Westfield College, Mile end road, London E1 4NS, England - 1992