MOIA - Deuxième Partie : Contraintes - Système Expert - Jeux

F. Bouquet

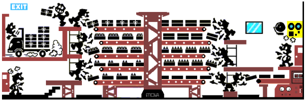
Master S&T - Mention Informatique

Première année

Programmation avec contraintes

	S	E	N	D
+	M	O	R	E
M	O	N	E	Y

- **Motivations :**
  - Comment déduire des informations pertinentes
  - Comment éviter les boucles infinies
  - Comment trouver la bonne heuristique
  - Comment faire la bonne abstraction



Programmation par contraintes

- **But :**
  - Définir les contraintes par rapport aux variables
  - Chercher une solution satisfaisant toutes les contraintes
- **Contraintes :**
  - Réduire le domaine des variables
  - Relation entre les inconnues
- **Caractéristiques :**
  - Information partielle : $X > 2$
  - Hétérogénéité : $N = \mathrm{Card}(S)$
  - Indirectionnelle : $X = Y + 2 \Leftrightarrow X \leftarrow Y + 2 \vee Y \leftarrow X - 2$
  - Déclaration Additive : $X > 2, X < 5 \Leftrightarrow X < 5, X > 2$
  - Rarement indépendante : $A + B = 5, A - B = 1$

CSP

Problème de Satisfaction de Contraintes
Constraint **S**atisfaction **P**roblem.

- **Un quadruplet** (X, D, C, R) :
  - $X = \{x_1, ..., x_n\}$, un ensemble de n **variables**
  - $D = \{D_1, ..., D_n\}$, un ensemble de **domaines** avec $x_i \in D_i$
  - $C = \{C_1, ..., C_m\}$, un ensemble de m **contraintes**
  - A chaque contrainte C_i est associée une **relation** R_i

Propriétés sur les contraintes

- Satisfiable : a-t-elle une solution ?

$$X = 1, X = Y + 1$$

- Non-Satisfiable :

$$X \leq 3, Y = X + 1, Y \geq 6$$

- Équivalentes si elles ont le même ensemble de solutions :

$$X = Y + 1, Y \geq 2 \leftrightarrow X = Y + 1, X \geq$$

- C_1 Implique C_2 : Si les solutions de C_1 est un sous-ensemble de C_2

$$\text{cont}(X, X) = \text{cont}(Y, nil) \rightarrow Y = nil$$

Méthodes de résolution

- Générer et tester
- Retour en arrière (Backtracking)
- k-Consistance
- Retour Arrière guidé (Look back)
- Avancement guidé (Look Ahead)
- Optimisations de contraintes
- Branchement par pondération (Branch and Bound)

Générer et Tester

- Méthode de résolution générale
- Systématique
- Algorithme :
  - Génération de valeurs (Labelling)
  - Test de la satisfaction

Inconvénient

- Génération aveugle
- Inconsistance découverte au dernier moment

Amélioration

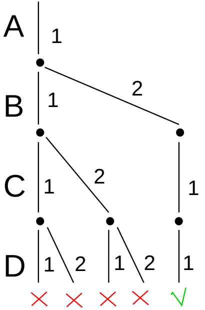
- Générateur intelligence
  $\rightarrow$ Recherche locale
- Tester dans le générateur
  $\rightarrow$ Backtracking

Backtracking

- Étendre incrémentalement une solution partielle en solution

$\boxed{A = D, B \neq D, A + C < 4}$

- **Algorithme :**
  - Répéter
  - Affecté une valeur à une variable
  - Tester la consistance
  - Jusqu'à ce que toutes les variables soient affectées
- **Inconvénients :**
  - Travail inutile
  - Travail redondant
  - Détection tardive de l'inconsistance



Exemple

- $X \in \{1,2\}, Y \in \{1,2\}, Z \in \{1,2\}$
- $X = Y, X \neq Z, Y > Z$

Générer et Tester

X	Y	Z	test
1	1	1	fail
1	1	2	fail
1	2	1	fail
1	2	2	fail
2	1	1	fail
2	1	2	fail
2	2	1	ok

Backtracking

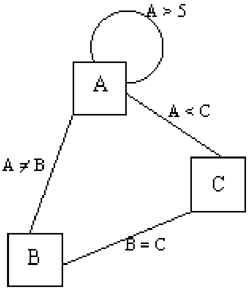
X	Y	Z	test
1	**1**	**1**	fail
1	1	**2**	fail
1	**2**		fail
2	**1**		fail
2	**2**	**1**	ok

Techniques de consistance

- Retirer les valeurs inconsistantes des domaines des variables
- Graphe représentant un CSP (uniaire, binaire)
  - nœuds ⇔ variables
  - arêtes ⇔ contraintes

Type de consistance :

- Consistance de nœuds (NC)
- Consistance d'arètes (AC)
- Consistance de chemin (PC)
- K-Consistance (Difficile)

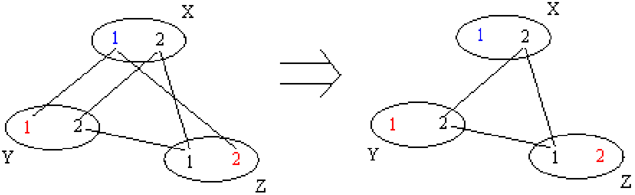


Consistance d'arètes (AC)

- Technique la plus communément utilisée
- Travail sur les contraintes binaires
- Variable ayant des supports

Exemple :

- $X \in \{1, 2\}, Y \in \{1, 2\}, Z \in \{1, 2\}$
- $X = Y, X \neq Z, Y > Z$



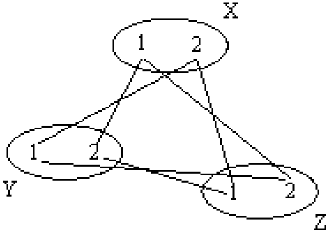
Consistance d'arètes (AC)

Dans quelle cas AC est suffisante :

- Domaine vide $\Rightarrow$ pas de solution
- Cardinalité de tous les domaines réduite à 1 $\Rightarrow$ Solution
- CSP équivalent
- Bon rapport : simplification / performance

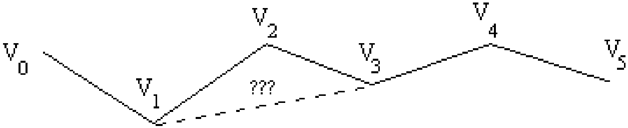
Problème :

- $X \in \{1, 2\}, Y \in \{1, 2\}, Z \in \{1, 2\}$
- $X \neq Y, X \neq Z, Y \neq Z$



Consistance de Chemins (PC)

- Consistance par rapport à un chemin



- Test un chemin de longueur deux
- Avantage / Inconvénient :
  + Détecte plus d'inconsistance que AC
  - Extension de la représentation des contraintes
  - Changement dans le graphe de connectivité

K-Consistance

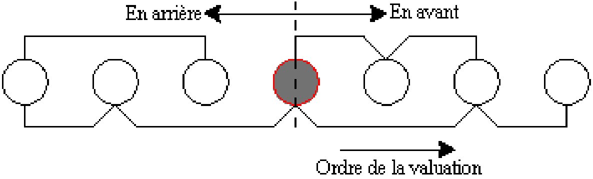
Définition *Un CSP est dit k-consistant si pour tout n-uplet de k variables* $(x_1, \ldots, x_k)$*, pour toute instanciation A consistances des* $k-1$ *variables* $(x_1, \ldots, x_{k-1})$*, il existe une valeur* $v \in d_k$ *telle que l'instanciation* $A \cup \{x_k = v\}$ *soit consitant*

Propriétés :

- Consistance de (K-1) variables pour la k^ième^ variable
- K-Consistance forte :
  $\equiv$ J-Consistance pour $J \leq K$
- AC : 2-Consistance forte
- PC : 3-Consistance forte

Propagation de contraintes

- Recherche systématique (GT & BT) seule : pas efficace
- Consistance seule : incomplète
- Combinaison des recherches (Backtracking) avec les techniques de consistance
- Méthodes :
  - Retour Arrière guidé (Look back) : Revenir avant le conflit
  - Avancement guidé (Look Ahead) : Prévenir les conflits

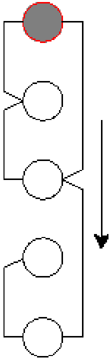


Avancement guidé

Prévenir de futur conflit

Deux niveaux :

- Partiel :
  - Forward Checking
- Complet :
  - Consistance d'Arc
  - Consistance de Chemin

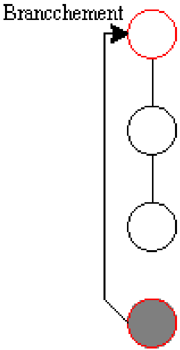


Forward Checking

- Filtrage sur les domaines
- Variables non instanciées
- Comparativement à la dernière valeur
- Instance courant consistance

Retour Arrière guidé

- Backtracking intelligent
- Consistance tester pendant l'instanciation des variables
- Retour avec saut (Backjumping) : Branchement au dernier conflit
- Backchecking $\sim$ Forward Checking
- Backmarking : Mémorisation



Exemple Problème du zèbre par Lewis Caroll

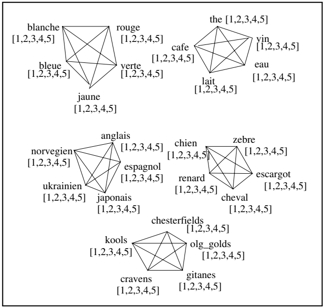
Cinq maisons consécutives, de couleurs différentes, sont habitées par des hommes de différentes nationalités. Chacun possède un animal différent, a une boisson préférée différente et fume des cigarettes différentes.

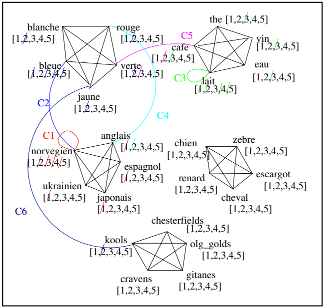
Qui boit de l'eau ? A qui appartient le zèbre ?

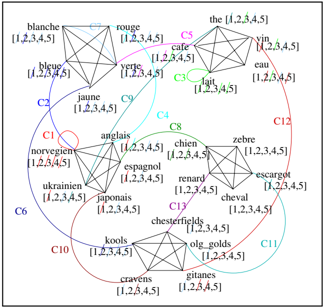
1. Le norvégien habite la première maison,
2. La maison à coté de celle du norvégien est bleue,
3. L'habitant de la troisième maison boit du lait,
4. L'anglais habite la maison rouge,
5. L'habitant de la maison verte boit du café,
6. L'habitant de la maison jaune fume des kools,
7. La maison blanche se trouve juste après la verte,
8. L'espagnol a un chien,
9. L'ukrainien boit du thé,
10. Le japonais fume des cravens,
11. Le fumeur de old golds a un escargot,
12. Le fumeur de gitanes boit du vin,
13. Le voisin du fumeur de Chesterfields a un renard,
14. Le voisin du fumeur de kools a un cheval.

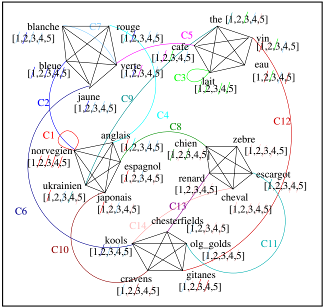
1. norvégien = 1,
2. bleue = norvégien + 1,
3. lait = 3,
4. anglais = rouge,
5. verte = café,
6. jaune = kools,
7. blanche = verte + 1,
8. espagnol = chien,
9. ukrainien = thé,
10. japonais = cravens,
11. old_golds = escargot,
12. gitanes = vin,
13. (chesterfields = renard + 1) ou (chesterfields = renard - 1),
14. (kools = cheval + 1) ou (kools = cheval - 1)

CSP - Zèbre


blanche [1,2,3,4,5]
rouge [1,2,3,4,5]
bleue [1,2,3,4,5]
verte [1,2,3,4,5]
jaune [1,2,3,4,5]
the [1,2,3,4,5]
vin [1,2,3,4,5]
cafe [1,2,3,4,5]
eau [1,2,3,4,5]
lait [1,2,3,4,5]
anglais [1,2,3,4,5]
norvegien [1,2,3,4,5]
espagnol [1,2,3,4,5]
ukrainien [1,2,3,4,5]
japonais [1,2,3,4,5]
chien [1,2,3,4,5]
zebre [1,2,3,4,5]
renard [1,2,3,4,5]
escargot [1,2,3,4,5]
cheval [1,2,3,4,5]
chesterfields [1,2,3,4,5]
kools [1,2,3,4,5]
olg_golds [1,2,3,4,5]
cravens [1,2,3,4,5]
gitanes [1,2,3,4,5]


blanche
rouge
bleue
verte
jaune
the
vin
cafe
eau
lait
anglais
norvegien
espagnol
ukrainien
japonais
chien
zebre
renard
escargot
cheval
chesterfields
kools
olg_golds
cravens
gitanes
C1
C2
C3
C4
C5
C6


blanche
rouge
bleue
verte
jaune
the
vin
cafe
eau
lait
anglais
norvegien
espagnol
ukrainien
japonais
chien
zebre
renard
escargot
cheval
chesterfields
kools
olg_golds
cravens
gitanes
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13


blanche
rouge
bleue
verte
jaune
the
vin
cafe
eau
lait
anglais
norvegien
espagnol
ukrainien
japonais
chien
zebre
renard
escargot
cheval
chesterfields
kools
olg_golds
cravens
gitanes
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14

Système Expert (Principe)

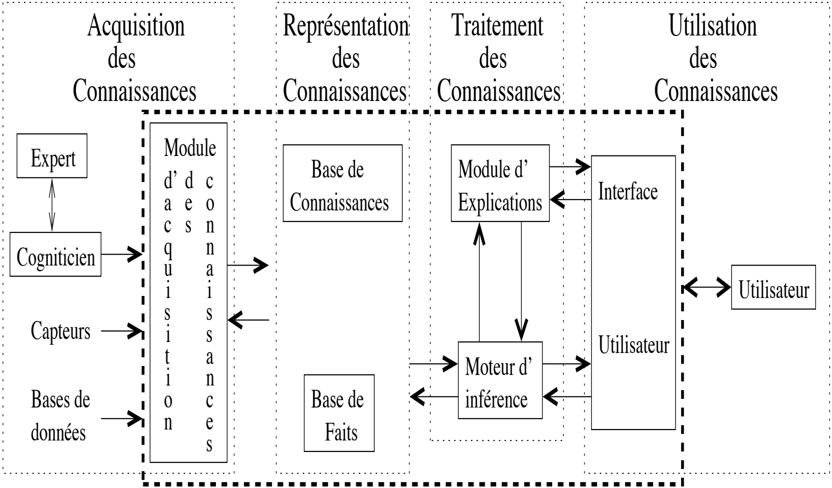
But :

- Modélisation d'un expert humain
- Tâche de résolution
- Explications sur les raisonnements

Composition :

- Base de connaissances
- Moteur d'inférence

Système Expert



Définition

Principaux éléments :

- Base de connaissances
- Moteur d'inférence
- Base de faits

Interface :

- Système de consultation
- Module d'explications
- Module d'acquisition des connaissances

Structures de contrôle

Cycle du moteur d'inférence :

- Phase de sélection
- Phase de filtrage
- Phase de résolution de conflits
- Phase d'exécution

Mode de raisonnements :

- Chaînage avant (données)
- Chaînage arrière (but)
- Chaînage mixte

Exemple

Soit la base de connaissances :

- R_1 : si B et D et E alors F
- R_2 : si D et G alors A
- R_3 : si C et F alors A
- R_4 : si B alors X
- R_5 : si D alors E
- R_6 : si A et X alors H
- R_7 : si C alors D
- R_8 : si X et C alors A
- R_9 : si X et B alors D

Base de faits : B, C

But : H

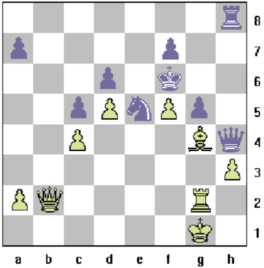
Chaînage avant

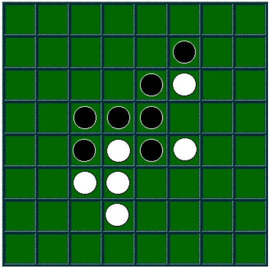
- Saisie des faits initiaux
- **Début**
  - Phase de filtrage ⇒ Détermination des règles applicables
  - **Tant que** ensemble règles applicables n'est pas vide **ET** que le problème n'est pas résolu **Faire**
    - Phase de choix ⇒ Résolution des conflits
    - Appliquer la règle choisie (exécution)
    - Modifier (éventuellement) l'ensemble des règles applicables
  - **Fin faire**
- **Fin**

Chaînage arrière

- Phase de filtrage
- **Si** l'ensemble des règles sélectionnées est vide **Alors**
  questionner l'utilisateur
- **Sinon**
  - **Tant que** le but n'est pas résolu **ET** qu'il reste des règles sélectionnées **Faire**
    - Phase de choix
    - Ajouter les sous-buts (partie gauche de la règle choisie)
    - **Si** un sous-but n'est pas résolu **Alors**
      mettre le sous-but en but à résoudre
  - **Fin faire**

Algorithmes pour les jeux





Minimax ou Négamax

- États, transitions : Arbre ou Graphe
- Deux joueurs $\not\Rightarrow$ Algorithmes type A^*

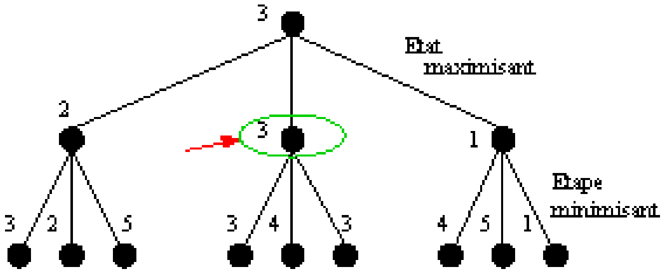
Composants :

- Fonction d'évaluation
- Profondeur de recherche
- Algorithme Minimax

$\Rightarrow$ Profondeur d'abord.

Exemple Minimax ou Négamax

- Vision double : Deux joueurs
- Évaluation par les feuilles



Propriétés Minimax

- *Complet :* Oui, si arbre de jeu fini
- *Optimal :* Oui, si adversaire est aussi optimal
- *Compléxité en temps :* $O(b^m)$
- *Compléxité en espace :* $O(bm)$

b nombre de branchements
m nombre de coups

$\alpha - \beta$

- Optimisation de Minimax
- Notion de seuil
- α : Pour un nœud Min *n*, valeur la plus grande de tous les nœud Max ancêtres de *n*

  Si val(n) $< \alpha$ exploration inutile
- β : Pour un nœud Max *n*, valeur la plus petite de tous les nœud Min ancêtres de *n*

  Si val(n) $> \beta$ exploration inutile

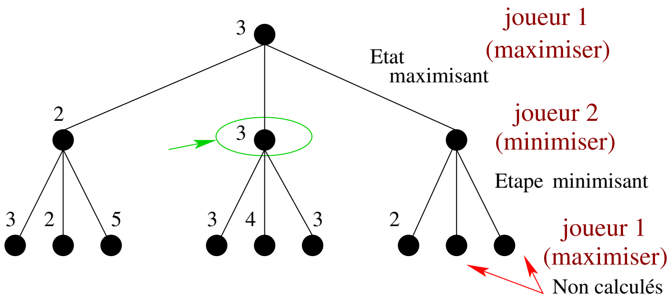
Algorithme AlphaBeta

Algorithme 1 : Algorithme : AlphaBeta(e,d, α, β)

Input : e état du système
Input : d profondeur
Input : $[\alpha, \beta]$ borne
if *n est terminal* **then return** h(n);
;
else if *n est de type Max* **then**
    Soit $(f_1, \ldots f_k)$ les fils de n;
    j $\leftarrow$ 1 ;
    while $j \leq k$ **et** $\alpha < \beta$ **do**
        $\alpha \leftarrow max(\alpha, AlphaBeta(f_j, \alpha, \beta))$;
        j $\leftarrow$ j + 1 ;
    return α ;
    else if *n est de type Min* **then**
        Soit $(f_1, \ldots f_k)$ les fils de n;
        j $\leftarrow$ 1;
        while $j \leq k$ **et** $\alpha < \beta$ **do**
            $\beta \leftarrow min(\beta, AlphaBeta(f_j, \alpha, \beta))$;
            j $\leftarrow$ j + 1
        return β

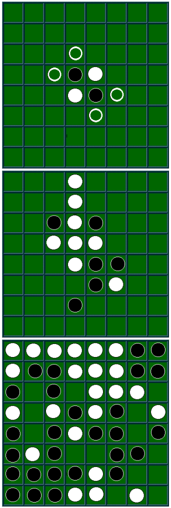
Exemple *AlphaBeta*

- Appel : $AlphaBeta(Racine, -\infty, +\infty)$
- Au mieux parcours $2\sqrt{N}$ vs N



Exemple - Othello

Borné à 60 demi-coups



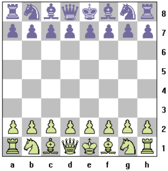
- **Phases de jeu :**
  - Début $\simeq$ *4 demi-coups* : Ouverture prés définie
  - Milieu : Valeur tactique variable

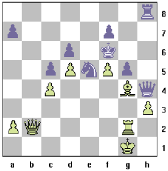
500	-150	30	10	10	30	-150	500
-150	-200	0	0	0	0	-200	-150
30	0	1	2	2	1	0	30
10	0	2	16	16	2	0	10
10	0	2	16	16	2	0	10
30	0	1	2	2	1	0	30
-150	-200	0	0	0	0	-200	-150
500	-150	30	10	10	30	-150	500

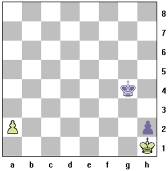
  - Fin $\simeq$ *15 demi-coups* : Calcul par différence

Exemple - Échec

10 à +100 demi-coups





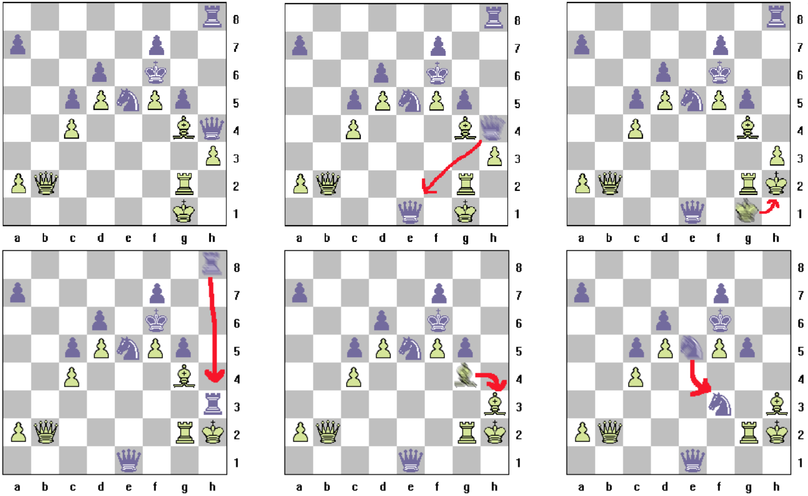


- **Phases de jeu :**
  - Début $\simeq$ *6 demi-coups* :
    Ouverture prés définie
  - Milieu :
    Valeur tactique

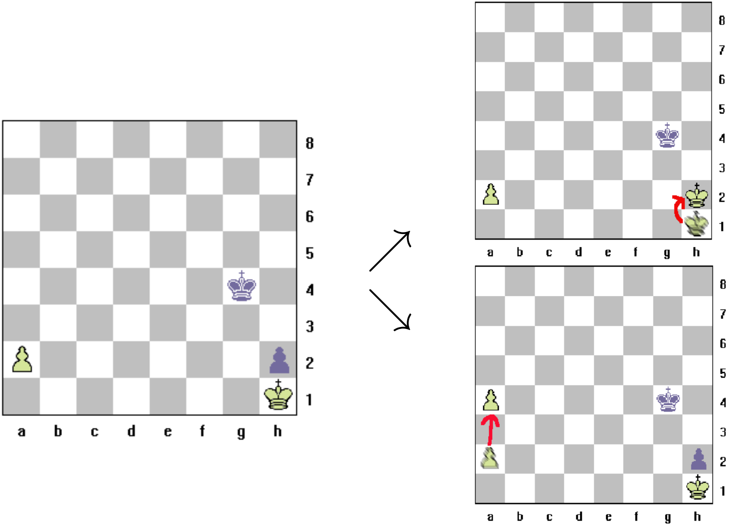
Dame	Tours	Fou	Cavalier	Pion
9	5	3	3	1

  - Fin $\simeq$ *8 pièces*

Effet de bloc



Effet d'horizon



Contrée

- *Annonces :*

Annonce	80	90	100	110	120	130	140...
Jeux	2As, V/9	2As, V&9	MA 4C	MA 3C	MA 2C	1 pli	

  Ajout : +10 pour 9, +20 pour Valet, +10 par As, +10 par dix second ou filante...

- La *partie :*
  - Règles : Si aller alors Jouer atout, Jouer ses as
  - Probabilité : Déduction lors des annonces (Coefficient mis à jour pendant le jeu)