Rapport du Projet - Secret Number Dilemma

Rousée Tom 22200898

1. Vue d'ensemble du projet

Secret Number Dilemma est un système de résolution intelligente pour un jeu de déduction de nombres secrets entre plusieurs joueurs. Le projet utilise une approche CSP (Constraint Satisfaction Problem) pour modéliser et résoudre le problème de manière optimale.

Objectifs

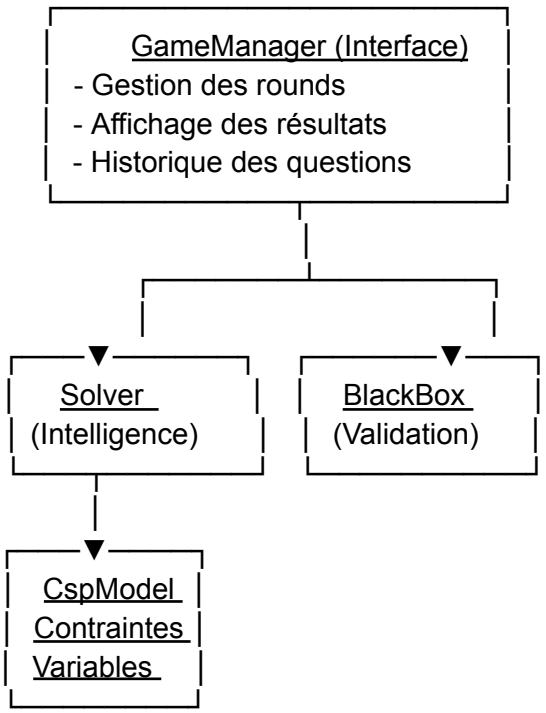
- Deviner le nombre secret de chaque adversaire (entre 0 et 100)
- Minimiser le nombre de questions posées
- Maximiser l'information gagnée tout en minimisant l'information révélée

---

2. Architecture Générale

2.1 Structure modulaire

Le projet suit une architecture en couches avec séparation claire des responsabilités :



2.2 Composants principaux

Solver (Intelligence artificielle)

- **Rôle** : Cerveau du système, évalue et sélectionne les meilleures questions
- **Responsabilités** :

    - Calcul de l'information gagnée et de la fuite d'information
    - Simulation des conséquences de chaque question
    - Gestion des connaissances sur les adversaires (`givenDomains`)

CspModel (Moteur de contraintes)

- **Rôle** : Gère et applique les contraintes CSP
- **Responsabilités** :
    - Application des contraintes sur les domaines
    - Simulation de contraintes sans modification
    - Propagation des contraintes

Variables (Représentation des domaines)

- **Rôle** : Représente l'espace de recherche pour chaque joueur
- **Responsabilités** :
    - Stockage des valeurs possibles (domaine)
    - Suppression de valeurs incompatibles
    - État actuel de la connaissance

---

3. Stratégie de Résolution

3.1 Approche CSP (Constraint Satisfaction Problem)

Le problème est modélisé comme un CSP où :

- **Variables** : Le nombre secret de chaque joueur
- **Domaine initial** : [0, 100] pour chaque variable
- **Contraintes** :
    - AllDifferent (tous les nombres sont différents)
    - Contraintes binaires issues des réponses aux questions

Le modèle CSP joue un double rôle essentiel dans la stratégie de résolution :

1. **Recherche de la solution (Déduction)** : Le CSP permet d'appliquer les contraintes (initiales comme AllDifferent et celles dérivées des réponses aux questions) pour réduire l'espace des domaines de valeurs possibles. Lorsqu'un domaine est réduit à une seule valeur, la solution pour ce nombre secret est trouvée. C'est le moteur de déduction du système.
2. **Simulation des questions (Évaluation)** : Avant de poser une question, le Solver utilise le `CspModel` pour *simuler* l'application des contraintes pour *chaque* réponse possible. Cela permet de calculer la "Réduction moyenne" et l'"Écart-type" (voir Critères d'évaluation), en évaluant l'impact informatif potentiel de la question sans modifier l'état réel du système.

3.2 Types de contraintes implémentées

AdditiveConstraint (Opération SUM)

Résultat = val1 + val2

Si résultat <= 20 → retourne 20

Si résultat >= 180 → retourne 180

Sinon → retourne résultat réel

MultiplicativeConstraint (Opération MUL)

Résultat = (val1 * val2) % 10

Retourne le dernier chiffre du produit

DivisionConstraint (Opération DIV)

Résultat = max(val1, val2) / min(val1, val2)

Division entière du plus grand par le plus petit

ZeroConstraint (Opération ZEROCOUNT)

Compte le nombre de zéros entre min(val1, val2) et max(val1, val2)

Exemple : entre 38 et 105 → 40, 50, 60, 70, 80, 90, 100 → 8 zéros (100 compte pour 2)

AllDifferenceConstraint

Contrainte globale : tous les nombres doivent être différents

Propagation : si une variable a une seule valeur, cette valeur est retirée des autres domaines

3.3 Algorithme d'évaluation des questions

Le Solver utilise un système de **scoring multi-critères** pour évaluer chaque question possible :

Formule de score

Score = (réduction_moyenne × W_REDUCTION)

- (écart_type × W_STDDEV)

- (fuite_information × W_LEAKAGE)

Avec :

W_REDUCTION = 0.7 (poids de la réduction du domaine)

W_STDDEV = 0.2 (poids de l'uniformité)

W_LEAKAGE = 0.1 (poids de la fuite d'information)

Critères d'évaluation

1. Réduction moyenne (`maxReduction`)

- Mesure la diminution attendue de la taille totale des domaines
- Calcul pondéré par la probabilité de chaque réponse possible
- Plus la réduction est importante, meilleur est le score

2. Écart-type (`standardDeviation`)

- Mesure l'uniformité de la réduction entre les différentes réponses possibles
- Un faible écart-type signifie que toutes les réponses apportent une information similaire
- Évite les questions "tout ou rien" (très informatives dans un cas, inutiles dans l'autre)

3. Fuite d'information (`infoLeakage`)

- Estime l'information révélée à l'adversaire
- Utilise la structure `givenDomains` pour maintenir une vision de ce que l'adversaire sait
- Calcule la réduction du domaine de l'adversaire en fonction de notre réponse

Les questions sont triés pour le compromis

```
+-------------------------------------------------------+
|      BEST QUESTIONS (from best to worst)              |
+-------------------------------------------------------+
 > 1. Any player + (sum)          | red: 66 sd: 45 leak: 66 score:   30 * RECOMMENDED
  2. Any player * (product)       | red: 47 sd: 41 leak: 47 score:   19
  3. Any player 0 (zeros)         | red: 15 sd: 31 leak: 15 score:    2
  4. Any player % (quotient)      | red: 16 sd: 45 leak: 16 score:    0
```