Projet P-A.N.D.R.O.I.D.E.

Branch and Bound pour les diagrammes d'influence

PINAUD David & BIEGAS Emilie

Université Sorbonne Sciences

Janvier 2021

Table des Matières

1 Introduction **3**
1.1 Les diagrammes d'influence (ID) 3
1.2 Les LIMIDs 3
1.3 Limites des implémentations actuelles 3
1.4 But du projet 6

2 Explication de l'algorithme de Branch and Bound **6**

3 Délivrables du projet **6**

1 Introduction

1.1 Les diagrammes d'influence (ID)

Les diagrammes d'influence (ID) sont des graphes acycliques dirigés présentant trois types de nœuds. Les nœuds de *chance*, illustrés graphiquement par un ovale, représentent des variables aléatoire (dont le domaine est fini et non vide). Les nœuds de *décision*, illustrés graphiquement par un rectangle, représentent des variables de décision (dont le domaine est fini et non vide) tandis que les nœuds d'*utilité*, illustrés graphiquement par un losange, représentent une fonction d'utilité locale exprimant la préférence. Lorsqu'il y a plusieurs nœuds d'utilité, l'utilité totale en est la somme (c'est une décomposition partiellement additive de l'utilité).

Les arcs de ce graphe représentent une dépendance entre les nœuds et ont une signification différente en fonction du type de nœud à son extrémité. En effet, si un arc pointe vers un nœud de chance, cela représente une dépendance probabiliste; si il pointe vers un nœud de décision, il a un but informatif; enfin, si il pointe vers un nœud d'utilité, il représente une dépendance fonctionnelle.

Les IDs respectent deux hypothèses, une hypothèse de régularité (les décisions sont ordonnées dans le temps) et une hypothèse dite non-oubliant (chaque décision est conditionnée par toutes les observations de décisions antérieures). L'ensemble des instanciations des nœuds de chance pour les décisions antérieures d'une certaine désicion est appelé l'*historique*.

1.2 Les LIMIDs

Les *LImited-Memory Influence-Diagrams* (LIMIDs) sont des diagrammes d'influence qui assouplissent les deux hypothèses précédemment citées.

Tout d'abord, les LIMIDs assouplissent l'hypothèse non-oubliant de manière à conditionner une décision sur un nombre limité d'observations et de décisions antérieurs pertinentes (pour un compromis qualité/complexité).

Ensuite, assouplir l'hypothèse de régularité permet par exemple de modéliser la coopération de problèmes de décision multi-agents où un agent n'est pas au courant de décision d'autres agents. Les IDs forment alors un sous-ensemble des LIMIDs.

1.3 Limites des implémentations actuelles

Les méthodes de résolution exactes actuelles de LIMIDs ou IDs sont limités par leur complexité en temps et en espace. Pour exemple, considérons un ID qui modélise un robot dans un labyrinthe représenté par une matrice $n \times m$ composé de cases *mur*, de cases *libre* et d'une case *objectif*. On représente le labyrinthe comme ci-dessous, les cases grises, blanches, et étoilés étant respectivement les cases *mur*, *libre*, et *objectif*.

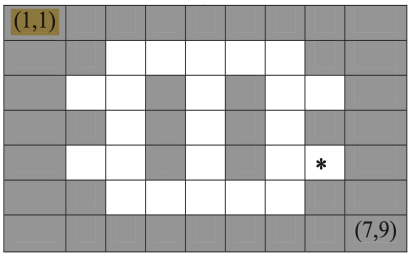

Figure 1: Un exemple de représentation d'un labyrinthe 9×7

Le robot est initialement placé sur une des cases *libre* et l'objectif du robot est d'atteindre la case *objectif*. À chaque étape, le robot peut se déplacer dans toutes les directions (nord, sud, est, ouest et les diagonales) d'une seule case ou choisir de ne pas se déplacer. Il possède quatre capteurs pointés vers les quatre cardinaux qui indiquent au robot la présence d'un mur ou non.

À chaque étape, le robot choisit une direction cardinale où faire un pas puis le mouvement du robot suit une mesure de probabilité :

- Un pas vers la case voulu a une probabilité $pBouger$ de réussir.
- Échouer de bouger survient avec probabilité $pEchecBouger$.
- À chaque étape, il y a une chance que le robot fasse un mouvement erratique:
  - Faire un pas vers la droite ou vers la gauche (c'est-à-dire vers l'est ou vers l'ouest si son choix est de se diriger vers le nord), s'il est possible de le faire, survient avec une probabilité de $pCote$ dans chacun des deux cas.
  - Faire un pas en arrière (c'est-à-dire vers le sud si son choix est de se diriger vers le nord), s'il est possible de le faire, survient avec une probabilité $pArriere$.
- Un pas vers un *mur* a une probabilité de 0.
- Les autres probabilités sont normalisées afin de former une distribution de probabilité.

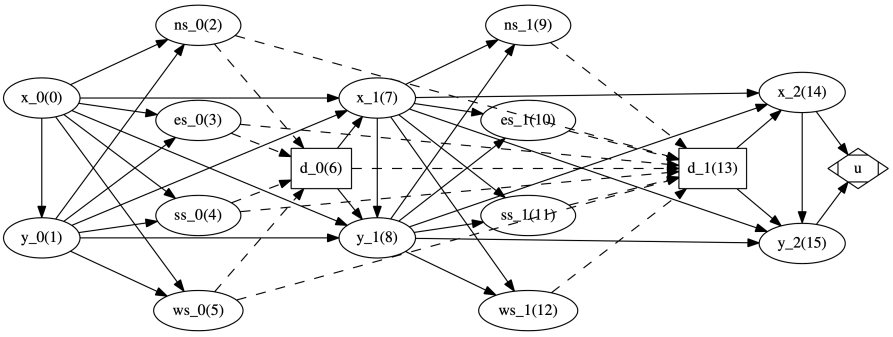

Figure 2: L'ID modélisant le problème du robot à deux étapes

A l'étape i, les nœuds de chance x_i et y_i représentent les coordonnées du robot sur la grille, ns_i, es_i, ss_i, ws_i les capteurs du robot dans les sens nord, est, sud et ouest respectivement. Les nœuds d_i sont les nœuds de décision.

Pour visualiser le caractère exponentiel de la complexité en espace, voici un tableau récapitulatif des tailles des arbres de jonctions selon le nombre d'étapes ainsi que les graphes associés.

Nombre d'étapes	Tree-width (largeur de l'arbre)	taille en mémoire de l'arbre (en Go)
2	11	2 E-3
3	15	38 E-3
4	20	787 E-3
5	24	12
6	29	984
7	34	76 E3
8	37	246 E3
9	42	19687 E3
109	46	315000 E3

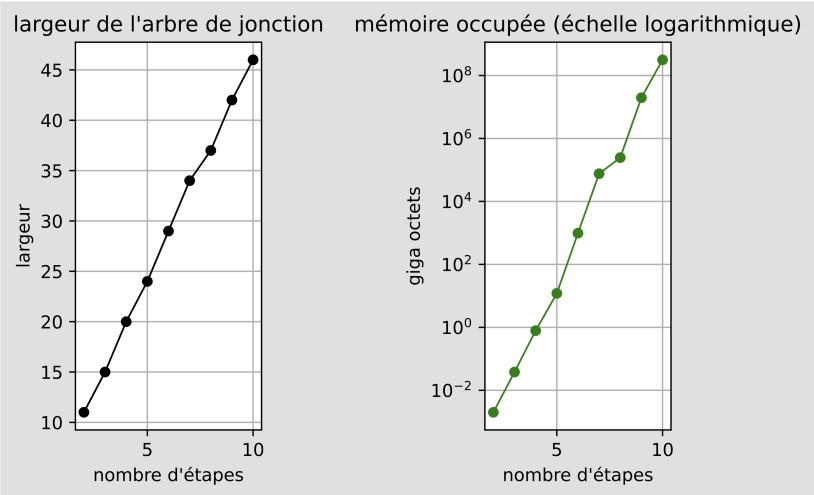

Figure 3: Graphes sur la complexité en temps et espace des méthodes actuelles

On parvient à résoudre cet ID avec les méthodes exactes actuelles (une machine avec 128Go de mémoire vive) sur des instances du problèmes comprenant jusqu'à 5 étapes. Au delà de 5 étapes, la mémoire nécessaire et le temps pour trouver la solution explose.

Il n'est pas envisageable de concevoir une machine capable de stocker autant de données pour résoudre de façon exacte cet ID. Il y a donc un réel besoin d'avoir une implémentation permettant de résoudre plus rapidement les IDs et en nécessitant moins d'espace.

1.4 But du projet

Le but de ce projet est donc d'étudier un algorithme de Branch and Bound pour résoudre des LIMIDs dans un petit graphe de recherche dans lequel différents chemins menant au même nœud représentent différentes histoires. Pour résoudre ce type de problème à grande complexité en espace et en temps, on doit tirer parti des opportunités pour le calcul de stratégie, utiliser des techniques d'inférence probabiliste et calculer les limites dans un graphe. Enfin, il est envisagé l'intégrer l'algorithme dans la librairie Python PyAgrum.

2 Explication de l'algorithme de Branch and Bound

Les LIMIDs sont résolus en trouvant une stratégie qui maximise l'utilité prévue. L'algorithme étudié dans ce projet résout les LIMIDs en les convertissant en un graphe ET/OU puis en effectuant un parcours en profondeur sur ces derniers. Les nœuds ET sont alors des variables aléatoires correspondant aux nœuds chances qui sont informatifs à un nœud de décision et les nœuds OU sont des nœuds de décision (représentant des alternatives de décision), enfin les feuilles sont les nœuds d'utilités. Un chemin racine-feuille représente alors une instance des nœuds d'information et de décision. En effectuant un parcours en profondeur d'abord, on peut générer l'arbre à la volée et ne garder qu'une partie de l'arborescence en mémoire. Deux problèmes se posent alors, le calcul des limites et celui des probabilités postérieures.

3 Délivrables du projet

Le projet a pour but de réaliser :

- Un état de l'art et stabilisation des algorithmes pour la résolution avec Branch and Bound, en particulier l'analyse du calcul des bornes et des probabilités postérieures.
- Une implémentation efficace et compacte des arbres ET/OU.
- Une intégration de l'algorithme correctement documentée (style python en anglais) dans l'API pyAgrum vérifiant l'implémentation des méthodes (liste non exhaustive) :
  - makeInference : Réalise l'inférence sur un LIMID.
  - PosterieurUtility : Retourne la probabilité postérieure d'un nœud.
  - lIMInfluenceDiagram : Retourne le LIMID associé à l'instance de la classe.
  - optimalDecision : Retourne la décision optimale pour un certain nœud de décision basé sur le critère MEU.
  - MEU : Retourne l'utilité maximate espérée de l'inférence.
  - junctionTree : Retourne l'arbre de jonction associé au LIMID
  - andOrTree: Retourne le graphe de jonction développé
  - une fonction pour avoir le graphe/arbre du branch and bound et possiblement une fonction pour avoir les états
- Une série de tests unitaires effectués sur les méthodes avec l'API TestUnit.

- Une série de benchmark validant l'utilité de notre implémentation de la méthode branch and bound face aux méthodes existante.