Université Toulouse III Paul Sabatier
Université de Toulouse

Rédigé avec LaTeX
Version du 2 octobre 2014

Techniques de base de résolution de problèmes

IA

M1 Informatique – Développement Logiciel
Semestre 7

Cours donné par Pierre Regner
Rédigé par Antoine de Roquemaurel

2014

Table des matières

1 Problèmes dans les espaces d'états

1.1 La réécriture

1.1.1 Algorithme en largeur d'abord

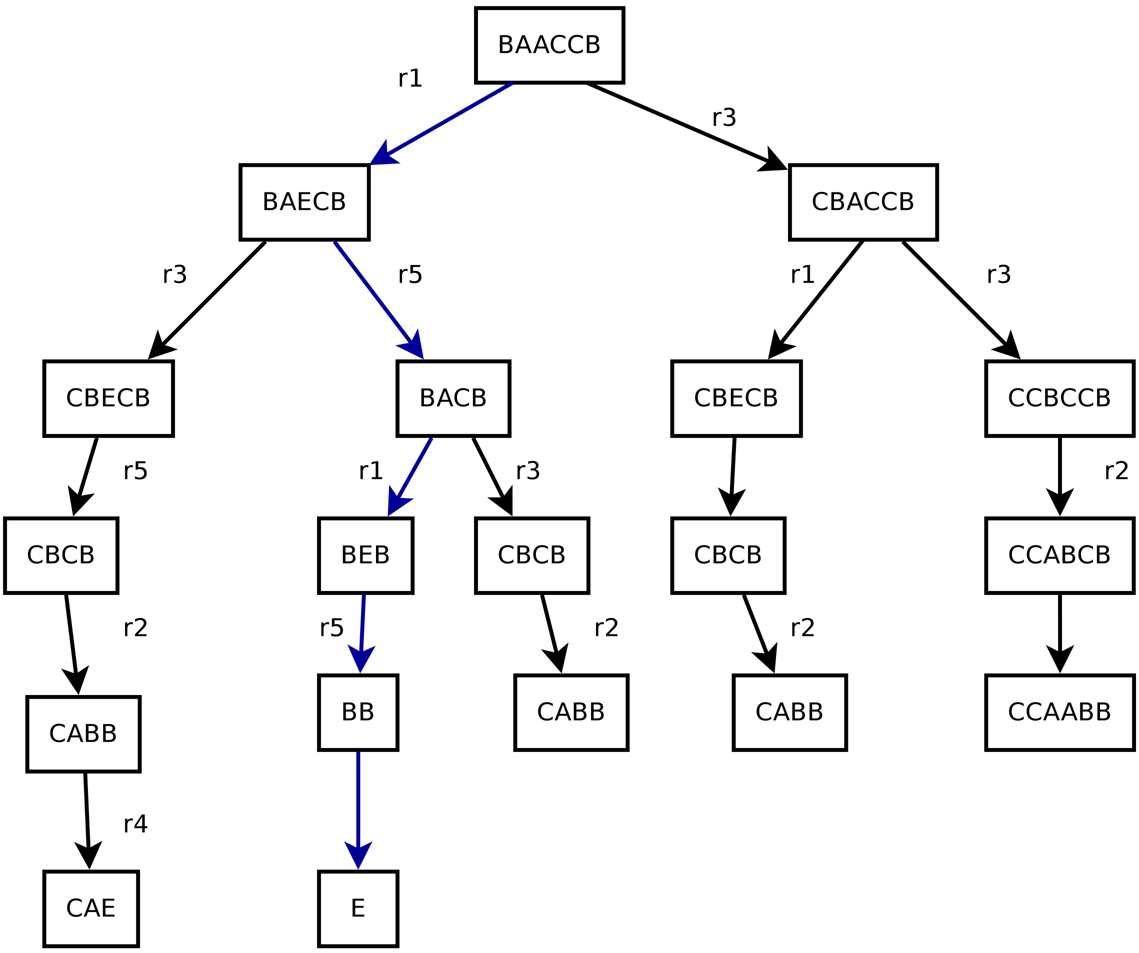


FIGURE 1.1 – Réécriture avec l'algorithme en largeur d'abord

Chemin solution `<r1, r5, r1, r5, r4>`

Nœuds développés 14

Nœuds crées 19

Algorithme optimal Longueur dans plans solution

Algorithme complet S'il existe une solution, il l'a trouve, sous condition de couper les branches déjà explorées

> **R** Cet algorithme consomme beaucoup de mémoire est peut être très long à dérouler : il n'est donc pas très utilisé

1.1.2 Algorithme en profondeur d'abord

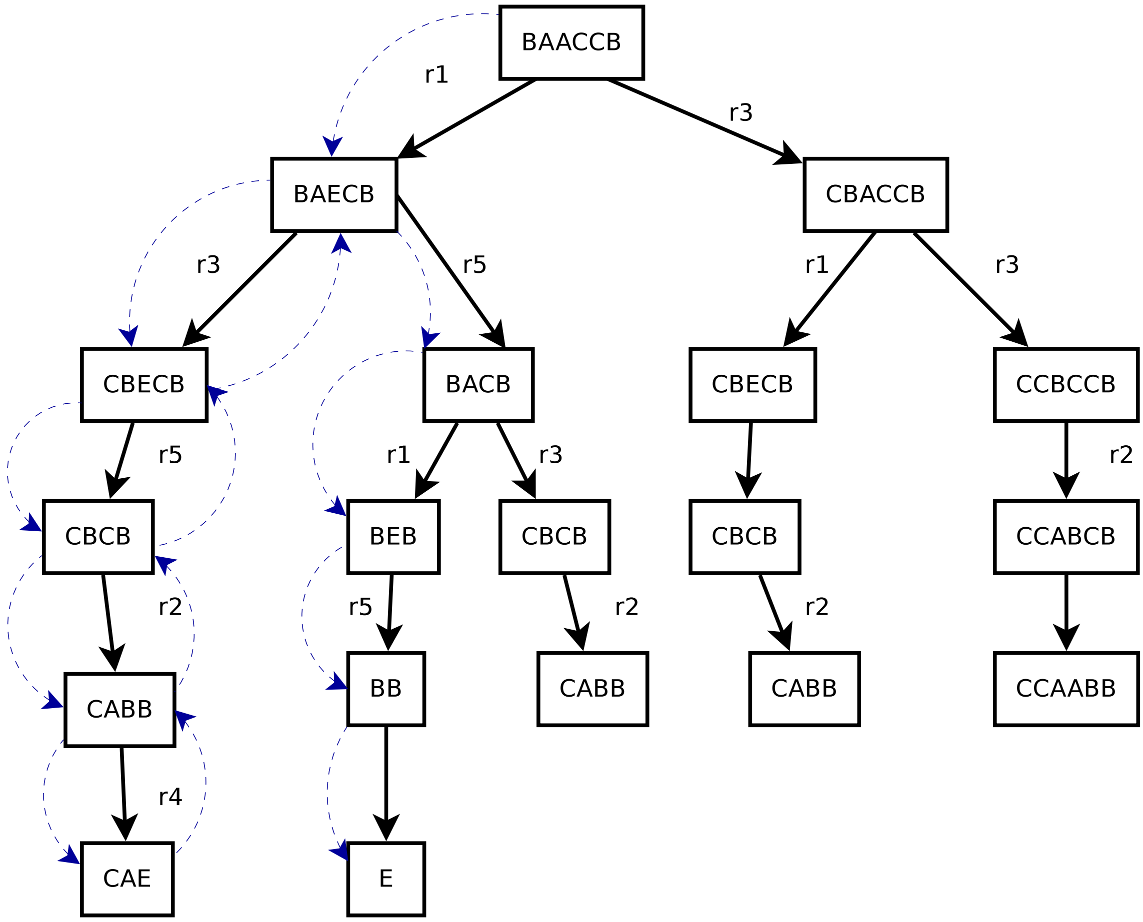


FIGURE 1.2 – Réécriture avec l'algorithme en profondeur d'abord

Chemin solution `<r1, r5, r1, r5, r4>`

Nœuds développés 8

Nœuds crées 10

Algorithme optimal Non optimal

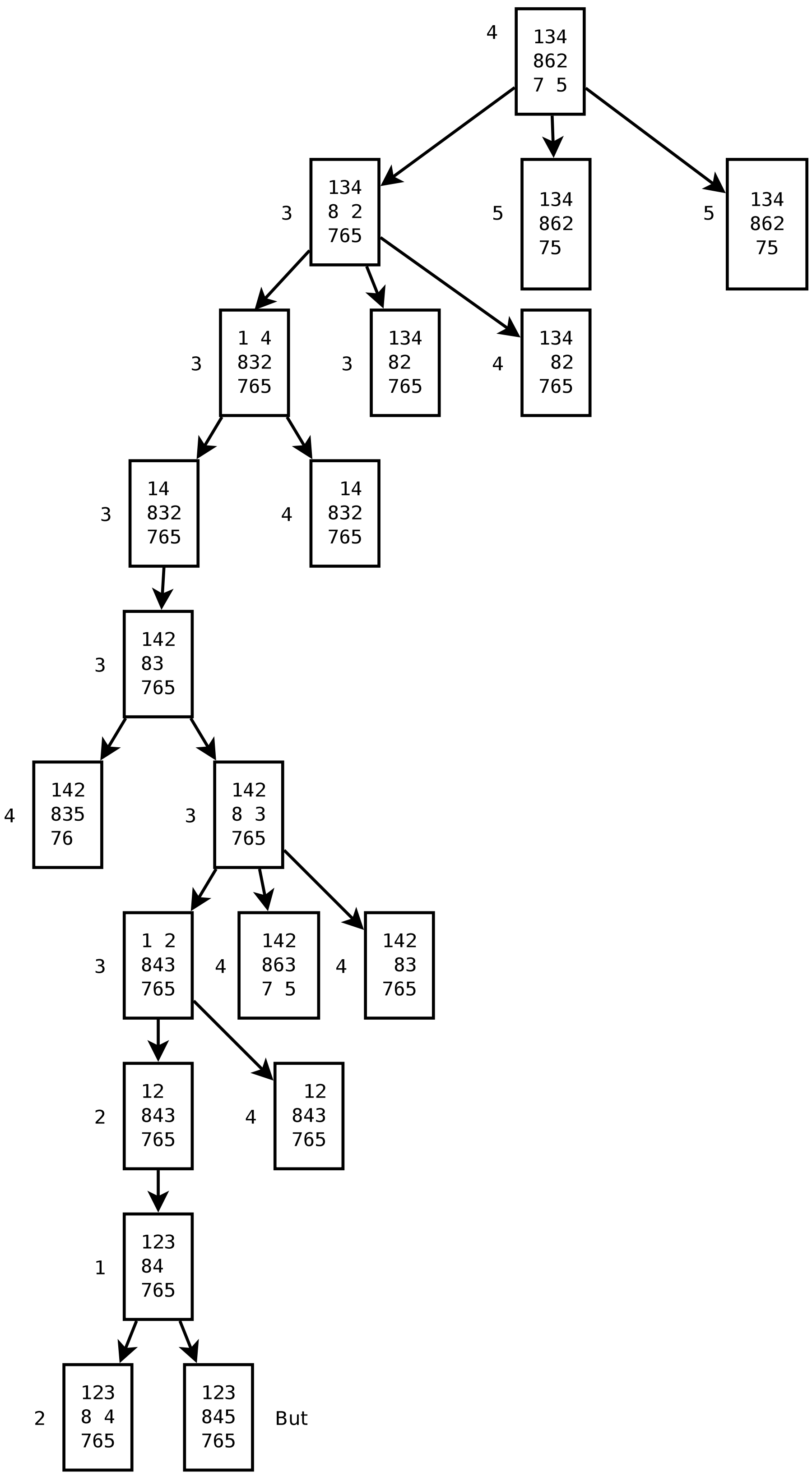
Algorithme complet S'il existe une solution, il l'a trouve, sous condition de couper les branches déjà explorées

> **R** Cet algorithme consomme beaucoup moins de mémoire que la profondeur d'abord

1.2 Le taquin 3×3

1.2.1 Algorithme « Glouton »

R L'algorithme Glouton peut aussi s'appeler Algorithme de Gradient



Heuristique Nombre de cases non en place

Opérateurs H,D,B,G

Chemin solution 9 étapes : `<H,H,D,B,G,H,D,B,G>`

Nœuds développés 9

Nœuds crées 20

1.2.2 Un autre problème

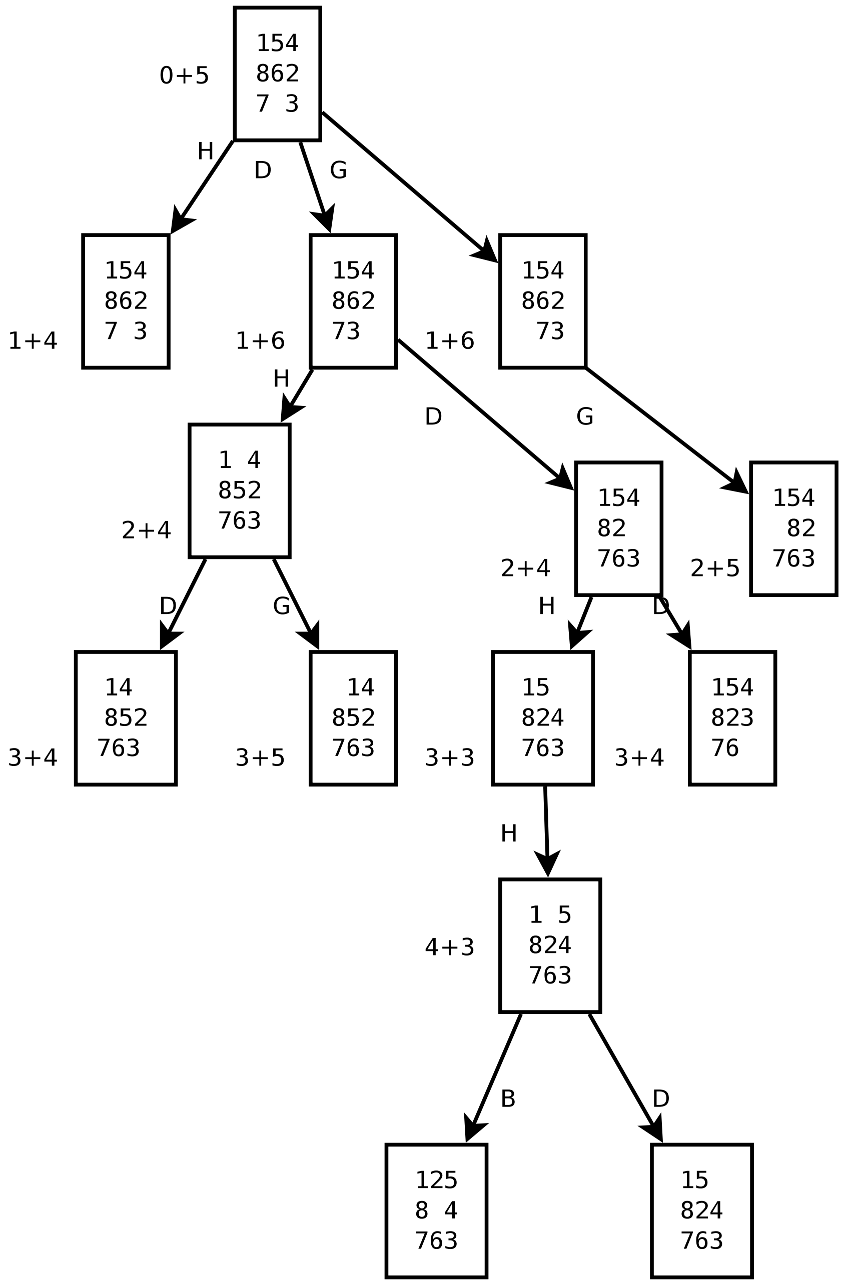


Figure 1.3 – Pas de solution

R Aucune solution, on est dans un autre espace d'état

1.3 Le jeu de cartes

On peut considérer les opérateurs prendre à droite/ prendre à gauche, mais on peut aussi conceptualiser un opérateur prendre 2 cartes ou seul la première compte : à ce moment là, notre arbre de recherche sera pratiquement deux fois moins long.

1.3.1 Modélisation des opérateurs

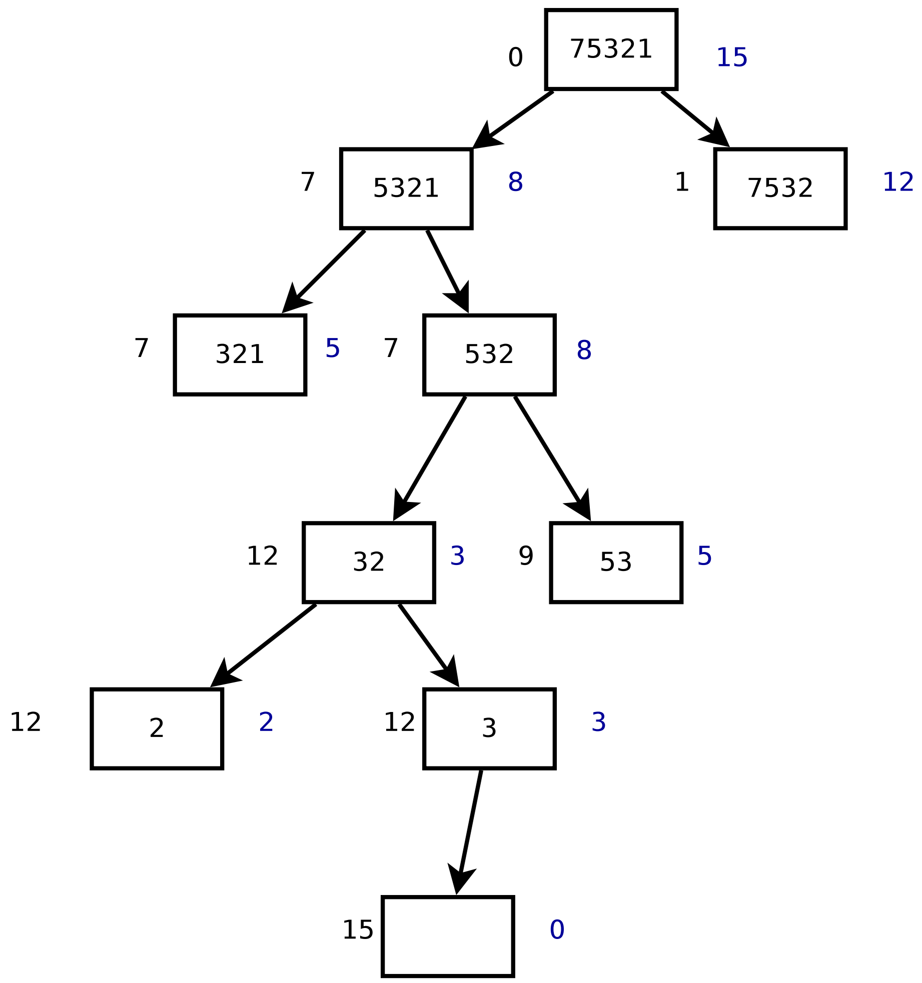
Solution à profondeur 5 :
- Prendre droite impair
- Prendre droite pair
- Prendre gauche impair
- Prendre droite impair

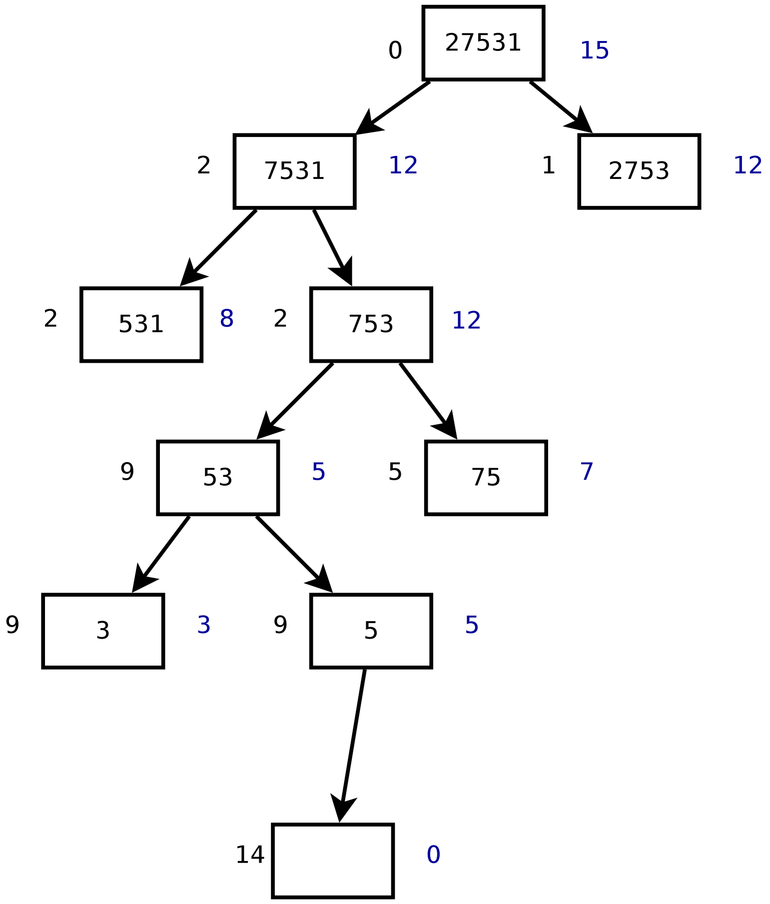
Solution à profondeur 3 :
- Prendre droite gauche
- Prendre droite droite
- Prendre gauche droite
- Prendre gauche gauche

C'est un problème de recherche dans les espaces d'états avec une optimisation des gains : c'est la différence entre un algorithme A et un algorithme A^*.

En théorie, afin d'avoir le meilleur gain, nous devrions parcours l'intégralité de l'arbre, cependant un algorithme nous évite de faire cela : l'algorithme «Meilleur d'abord»

On utilise une fonction d'estimation des états (heuristique) : la somme des n plus grandes valeurs de cartes avec n définis comme le nombre de cartes peuvent encore rapporter es gains. Cette fonction sur estime la réalité.





Notre heuristique est majorante : ainsi, il est inutile de backtraquer l'intégralité de l'arbre, les autres branches ne nous proposent pas plus que 14.

2 Arbres de jeux, stratégies, minimax

2.1 Arbre de jeu, stratégies

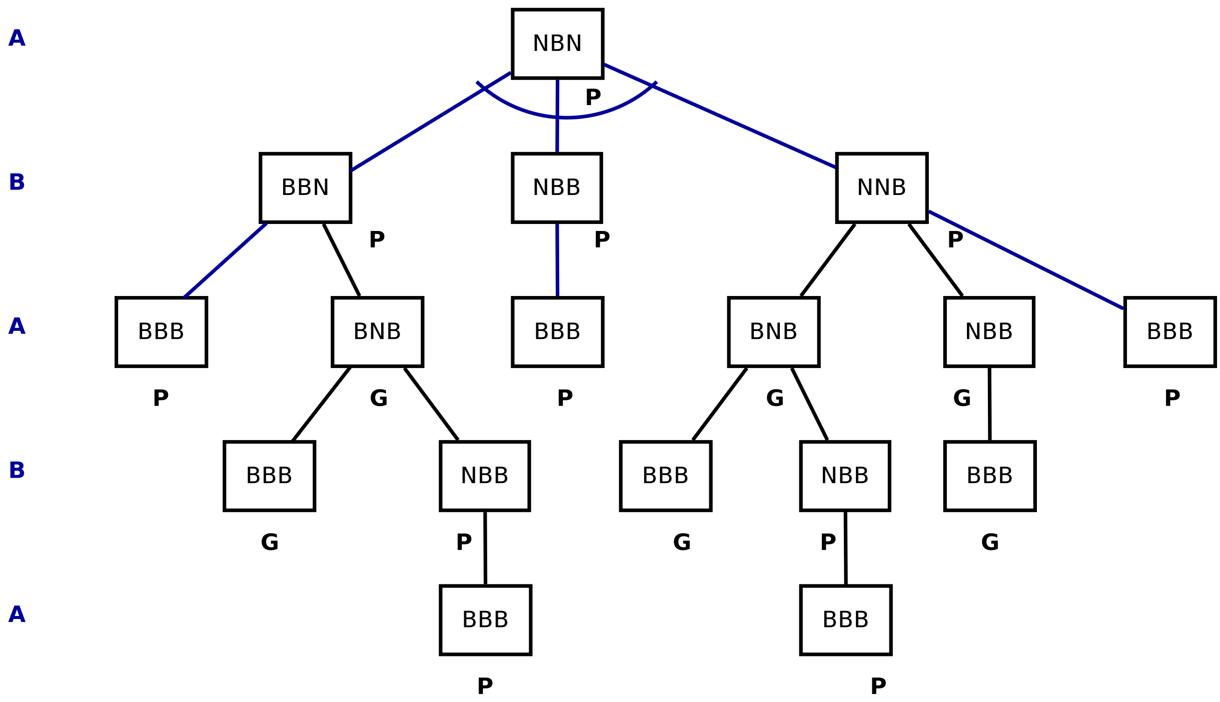


FIGURE 2.1 – Arbre de jeu

A commence, on choisit A comme joueur de référence, donc B est l'adversaire.

G A Gagne

P A Perd

E Égalité

2.1.1 Hypothèse

Chaque joueur joue du mieux possible !

2.1.2 Convention Minimax

— Le joueur de référence, A, remonte le maximum de ses fils
— L'adversaire, B, remonte le minimum de ses fils.

On pose $G > E > P$.

Cet arbre représente l'ensemble des parties possibles : une partie est une chemin partant de la racine pour aboutir à une feuille.

P remonté à la racine, A ne peut donc pas gagner : en bleu nous avons une stratégie gagnante pour B.

Definition 2.1 Une **stragégie gagnante** est un sous arbre de l'arbre de jeu tel que :
— La racine est identique
— Les nœuds du gagnant (celui qui possède une stratégie gagnante) sont les nœuds OU
— Les nœuds du perdant sont des nœuds ET.
— Pour chaque nœud OU on choisi une branche gagnante (pour le gagnant)
— Pour chaque nœud ET on retient toutes les branches
— Toutes les feuilles doivent être gagnantes (pour le gagnant)

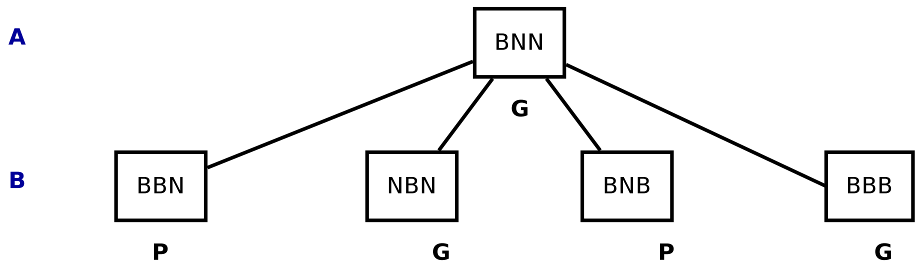


FIGURE 2.2 – Arbre de jeu

R La construction de cet arbre à été effectué à l'aide de l'arbre 2.1.

Dans ce cas là, il existe une stratégie gagnante pour celui qui commence

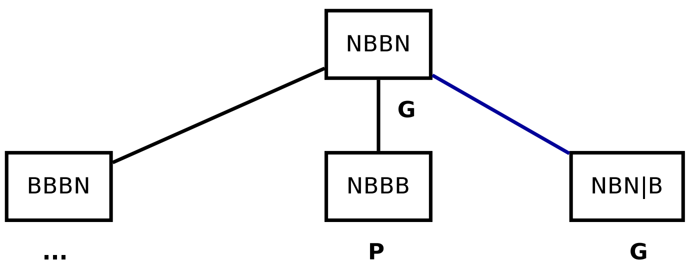


FIGURE 2.3 – Arbre de jeu

Il existe une stratégie gagnante pour A : inutile de développer le sous arbre gauche étant donné que nous savons qu'il en existe au moins une.

2.2 Minimax d'un arbre de jeu

Il existe une stratégie gagnante pour B, et le minimum des gaints possibles pour B, si celui joue bien, est de 1 (-1 en racine).

Si A joue mal, pendant que B conserve son jeu, alors B pourrait gagner plus, jusqu'à 17.

A Liste des codes sources