TP_5 Intelligence artificielle
Costa da Quinta, João Filipe

5.1 Formalisation

① variables du problème :

→ c'est toutes les zones à colorier

$X = (NB, PLCB, CA, RAA, MPLR, PACA, V)$

② Domaine des variables :

Soit C l'ensemble de toutes les couleurs

⇝ $\forall x_i \in X \quad D_i = C \setminus \{\text{couleurs des voisins de } x_i\}$

③ contraintes

Soit C l'ensemble des couleurs

$\langle NB, PLCB \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle BB, V \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle PLCB, V \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle RAA, V \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle PLCB, RAA \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle PLCB, CA \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle RAA, CA \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle CA, MPLR \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle RAA, MPLR \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle RAA, PACA \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

$\langle PACA, MPLR \rangle : \langle x, y \rangle$ tq $x, y \in C \wedge x \neq y$

⚠ pas de frontière entre Paca et V

Dans le .py elles seront représentées à l'aide d'une matrice $M(7,7)$ où $M(i,j) \in \{0,1\}$

si $M(i,j) = 0$ ⇝ pas de contrainte

else ⇝ contrainte

5.2 backtracking algorithm

pour découvrir le nombre min de couleurs, on essaye avec $n=1$ couleurs, et si on ne trouve pas, on essaye avec $n = n+1$ couleurs

① $C = \{rouge\}$ $|C| = 1$

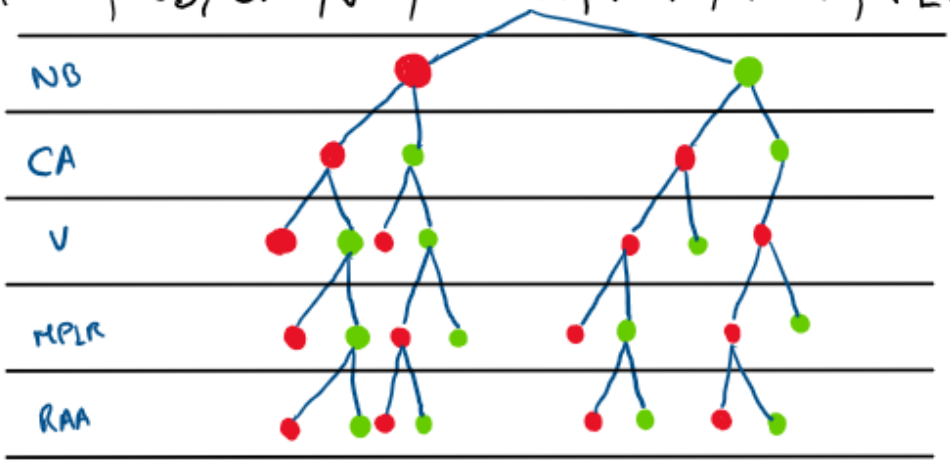
$Si = \{NB, CA, V, MPLR, RAA, PACA, PLCB\}$

NB → {NB : rouge} ✔ — CA → {CA : rouge} ✔ — V → {V : rouge} ✘.

on ne peut pas backtrack car le domaine pour toute variable affecté est vide.

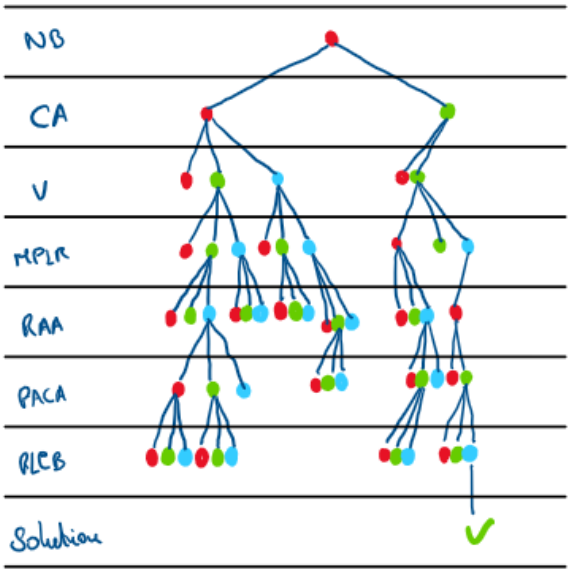
② $C = \{rouge, vert\}$ $|C| = 2$

$Si = \{NB, CA, V, MPLR, RAA, PACA, PLCB\}$



③ $C = \{rouge, vert, bleu\}$

$S_i = \{NB, CA, V, MPLR, RAA, PACA, PLCB\}$



$S_G = \{NB=$ ●, $CA=$ ●, $V=$ ●, $MPLR=$ ●, $RAA=$ ●, $PACA=$ ●, $PLCB=$ ●$\}$

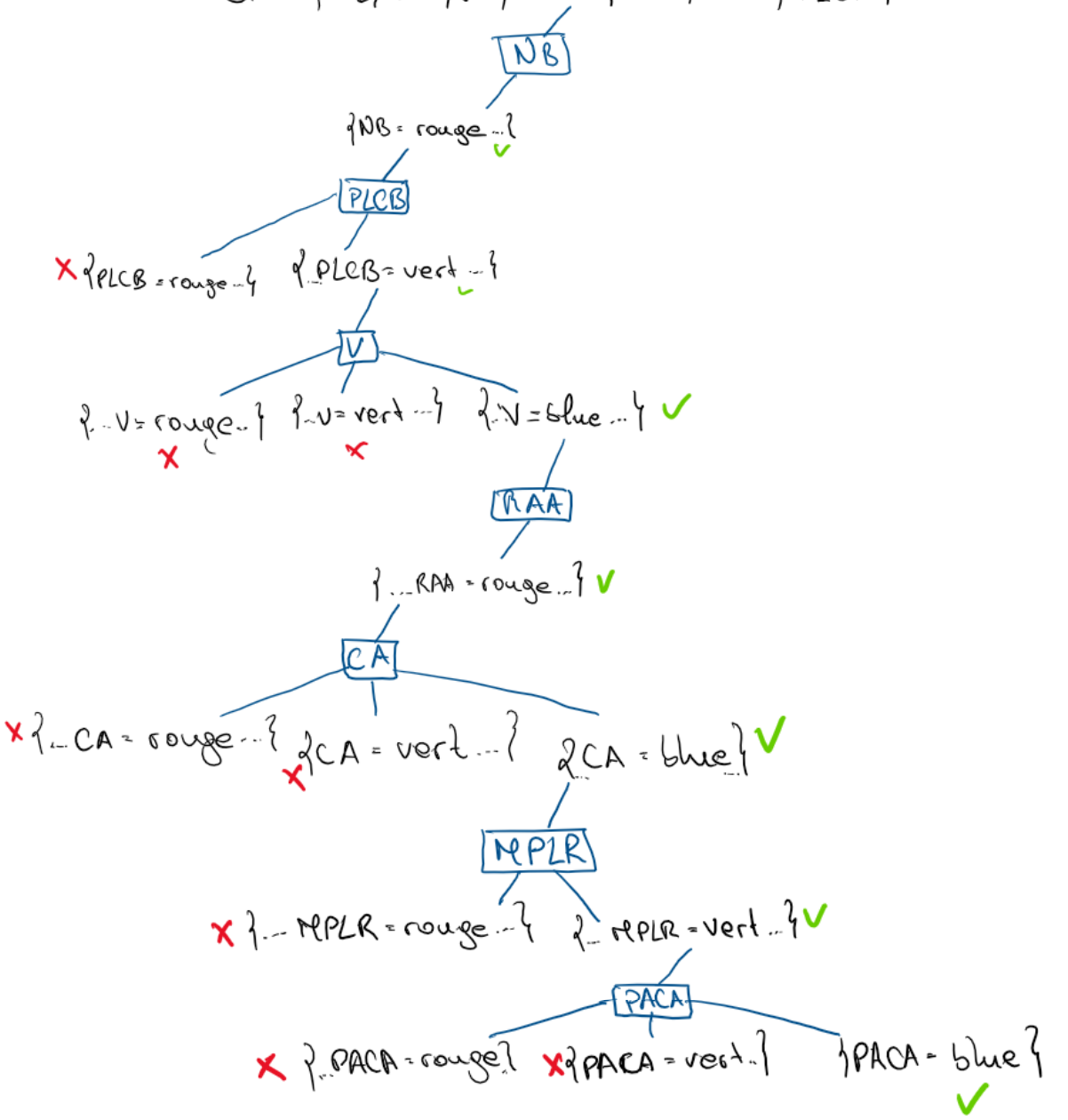
56 noeuds visités avant de trouver une solution.

↝ On a trouvé une solution avec $|C|=3$, on peut déjà remarquer que RAA c'est une région qui pose beaucoup de problèmes.

5.3 heuristique ⇝ variable la plus contrainte

$C = \{rouge, vert, blue\}$

$S_i = \{NB, CA, V, MPLR, RAA, PACA, PLCB\}$



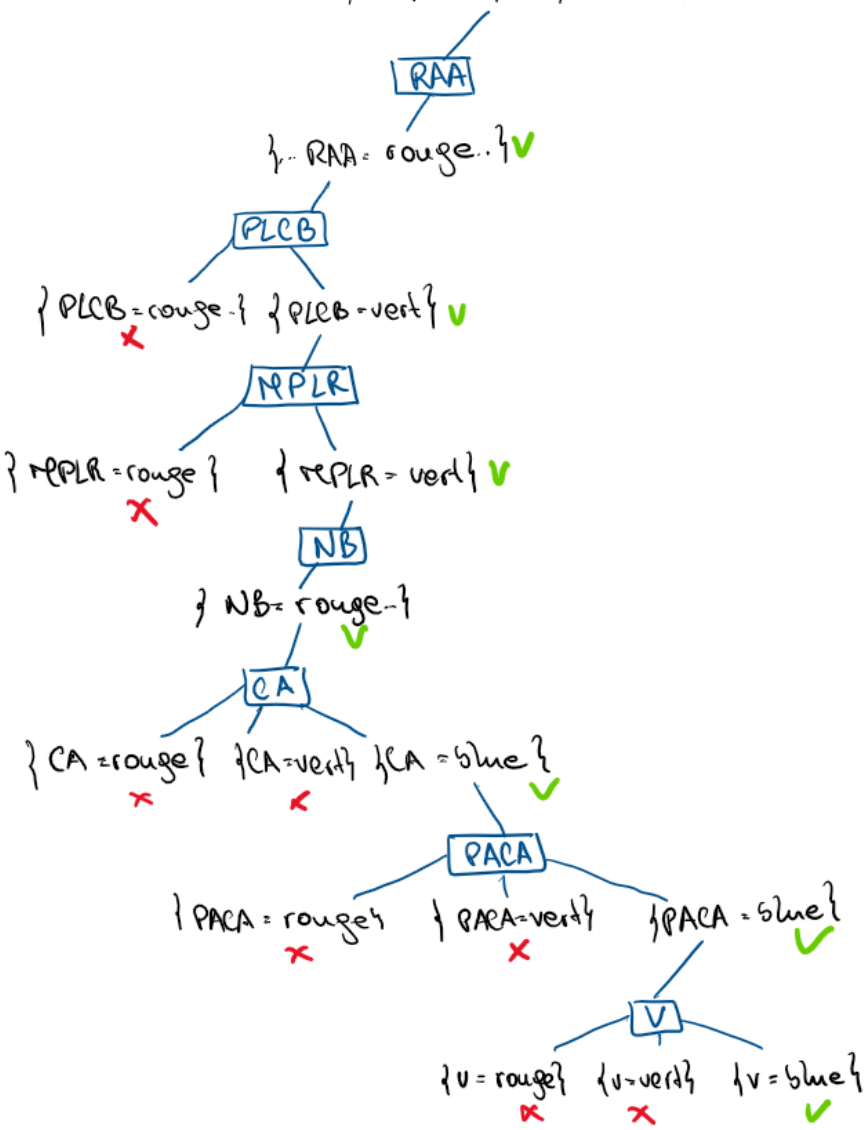
$S_G = \{NB = \bullet, PLCB = \bullet, V = \bullet, RAA = \bullet, CA = \bullet, MPLR = \bullet, PACA = \bullet\}$

15 noeuds visités avant de trouver une solution.

5.4 Heuristique → variable la plus contraignante

C = {rouge, vert, blue}

Si = {NB, CA, V, MPLR, RAA, PACA, PLCB}



SG = {RAA = ● , PLCB = ● , MPLR = ● , NB = ● , CA = ● , PACA = ● , V = ● }

15 noeuds visités avant de trouver une solution.

5.5 Choix de la bonne couleur

Vu que dans la 5.3 et 5.4 j'ai le même nombre de noeuds visités, je vais choisir comme heuristique région la 5.4 car le choix de la première région n'est pas aléatoire.

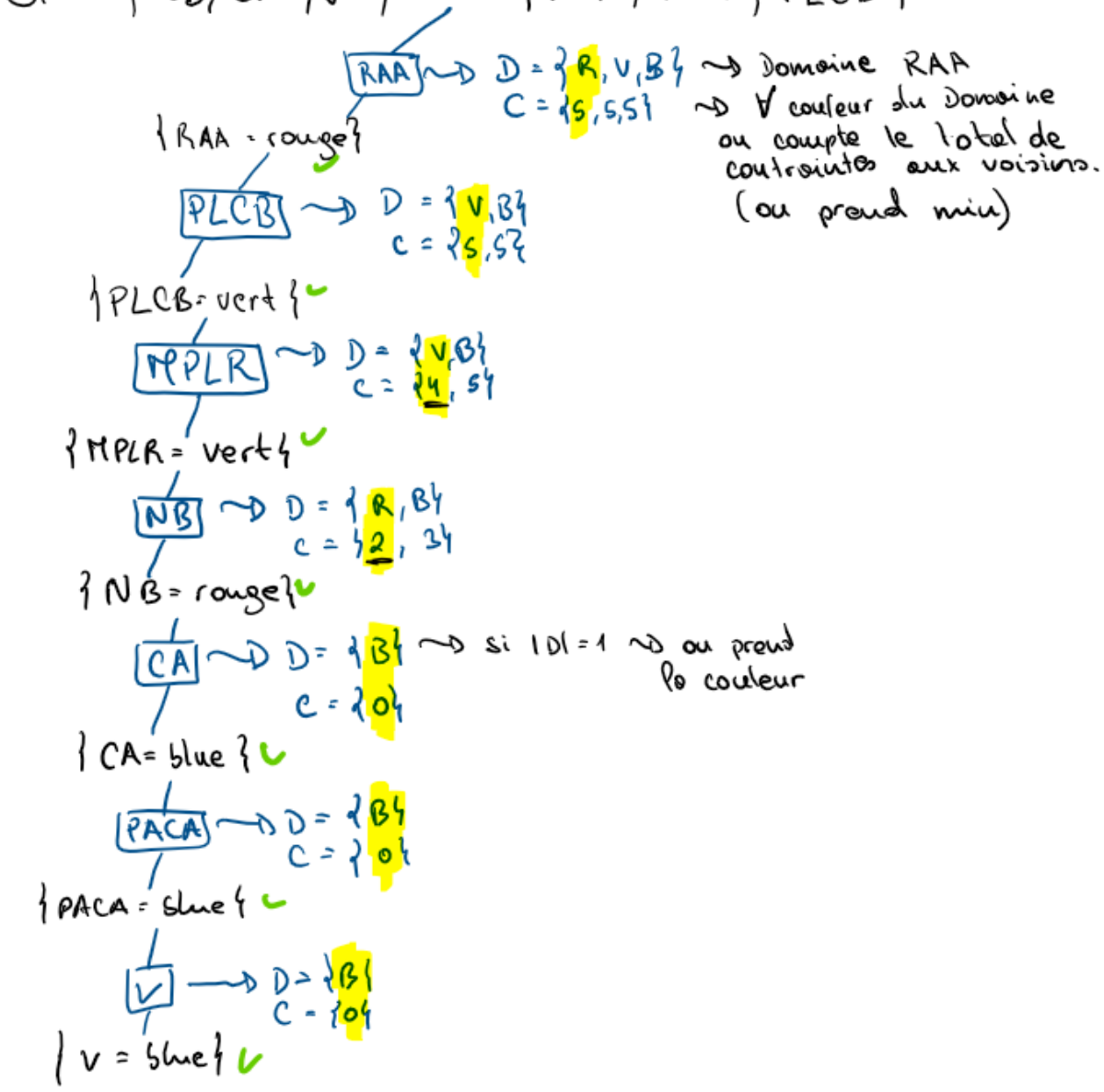
Et ensuite à chaque étape on choisit la couleur qui posera le moins de problèmes plus tard.

Pour choisir la bonne couleur on utilise l'algo suivant :

① Soit x la region à colorier
Soit y les regions voisines de x non coloriés
⇝ alors on choisit pour x la couleur de Dx qui imposera le moins de contraintes à y.

C = {rouge, vert, blue}

Si = {NB, CA, V, MPLR, RAA, PACA, PLCB}



SG = {RAA = 🔴, PLCB = 🟢, MPLR = 🟢, NB = 🔴, CA = 🔵, PACA = 🔵, V = 🔵}

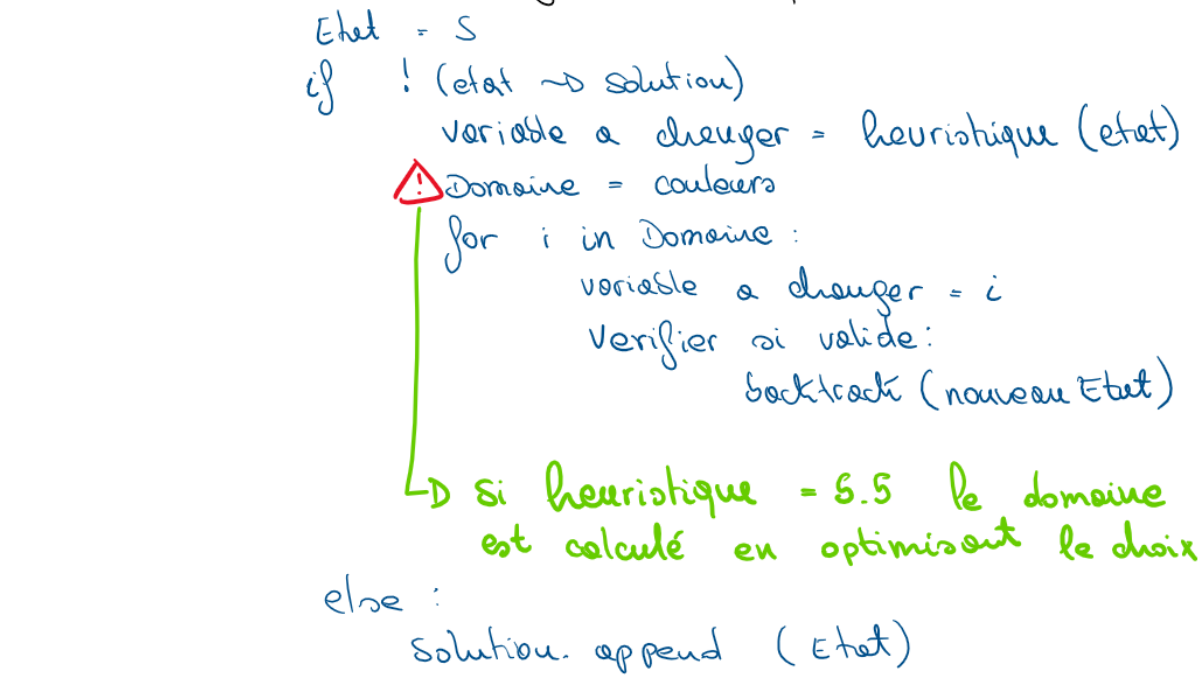
7 noeuds visités

5.6 Implementation

→ Code :

pour choisir l'heuristique il faut decomenter la bonne sur le code.

le code backtracking respecte le pseudo code suivant :



DANS LA FONCTION BACKTRACKING:

```
def backtracking(etatS):
    heuristique = "5.2"
    # heuristique = "5.3"
    # heuristique = "5.4"
    # heuristique = "5.5"
```

Fin du fichier .py