# Proposition de stage : Algorithmes de recherche pour la synthèse de programmes

Nathanaël Fijalkow  Guillaume Lagarde

---

**Encadrants** : Nathanaël Fijalkow et Guillaume Lagarde (CNRS, LaBRI, Bordeaux)

**Localisation** : LaBRI (Bordeaux)

**Sujet du stage** : Ce stage propose d'étudier le problème de la synthèse de programme d'un point de vue algorithmique.

**Thèmes** : algorithme, algorithme randomisé, synthèse

---

La motivation du problème étudié dans ce stage est la synthèse de programme : à partir d'une spécification, automatiquement construire un programme satisfaisant cette spécification. Ce problème à de nombreuses applications en pratique. Dans l'application concrète qui nous intéresse et a inspiré le cadre théorique que l'on décrit ci-dessous, nous présentons un exemple de programme en Figure 1.

```
a ← [int]                    An input-output example:
b ← SORT a                   Input:
c ← FILTER (>0) b            [-17, -3, 3, 11, 0, -5, -9, 13, 6, 6, -8, 11]
d ← HEAD c                   Output:
e ← DROP d b                 [-5, -3, 0, 3, 6, 6, 11, 11, 13]
```

Figure 1: Un exemple du type de programme synthétisé

Le cadre théorique dans lequel nous nous plaçons pour l'étude de la synthèse de programme est essentiellement un jeu de Mastermind : il s'agit de trouver la combinaison secrète. La subtilité est que la combinaison secrète est choisie selon une distribution dont nous avons des informations, typiquement les marginales : pour chaque couleur, nous connaissons la probabilité qu'elle apparaisse dans la combinaison secrète.

Pour illustrer les questions que l'on se pose, considérons le cas particulier suivant. Les programmes possibles ont deux lignes, et chaque ligne est $a$, $b$, ou $c$ (abstractions de fonctions plus compliquées). Il y a donc $9$ programmes possibles :

$$aa, ab, ac, ba, bb, bc, ca, cb, \text{ et } cc$$

La combinaison secrète est l'un de ces programmes. Un algorithme de recherche est simplement un ordre sur ces $9$ programmes, il décrit dans quel ordre les programmes sont testés.

Nous ne connaissons pas la distribution exacte selon laquelle la combinaison secrète est choisie, mais nous savons que ses marginales sont uniformes : la distribution $\mathcal{D}$ satisfait les propriétés suivantes, pour toute lettre $\sigma \in \{a, b, c\}$ et position $p \in \{1, 2\}$, la probabilité que $\sigma$ apparaisse en position $p$ est $\frac{1}{3}$. Il existe de nombreuses distributions satisfaisant ces propriétés,

exemple :

$$\mathcal{D}_1(aa) = \mathcal{D}_1(bb) = \mathcal{D}_1(cc) = \frac{1}{3} \text{ et } \mathcal{D}_2(t) = 0 \text{ sinon}$$

$$\mathcal{D}_2(ab) = \mathcal{D}_2(bc) = \mathcal{D}_2(ca) = \frac{1}{3} \text{ et } \mathcal{D}_2(t) = 0 \text{ sinon}$$

Étant donné un algorithme (c'est-à-dire un ordre sur les $9$ programmes) et une distribution, la complexité est l'espérance du nombre de programmes testés avant de trouver le bon programme. La question est : quel est le meilleur algorithme contre toutes les distributions ci-dessus (dites à marginales uniformes), et quelle complexité a t-il ?

## Objectifs du stage

L'objectif de ce stage est de développer de nouveaux algorithmes dans le cadre décrit ci-dessus, ainsi que dans plusieurs extensions motivés par la pratique. Selon l'avancement, le goût et l'intérêt du stagiaire, des implémentations peuvent être mises en œuvre.