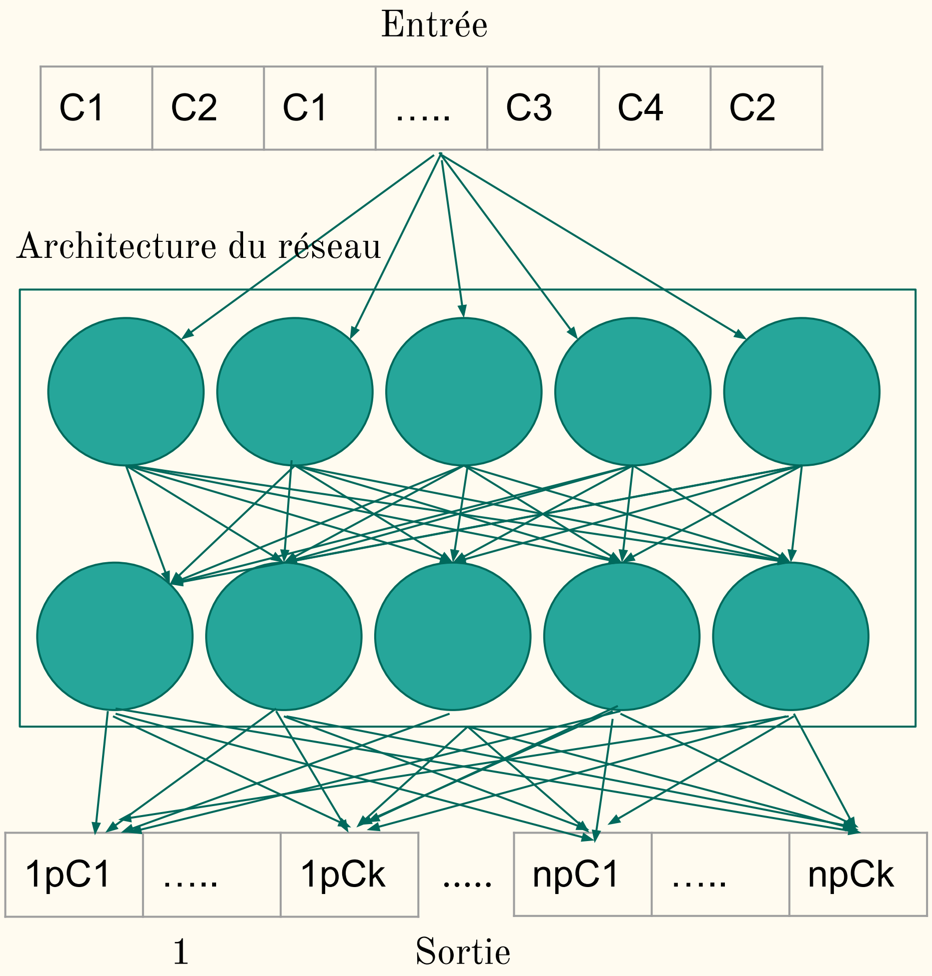
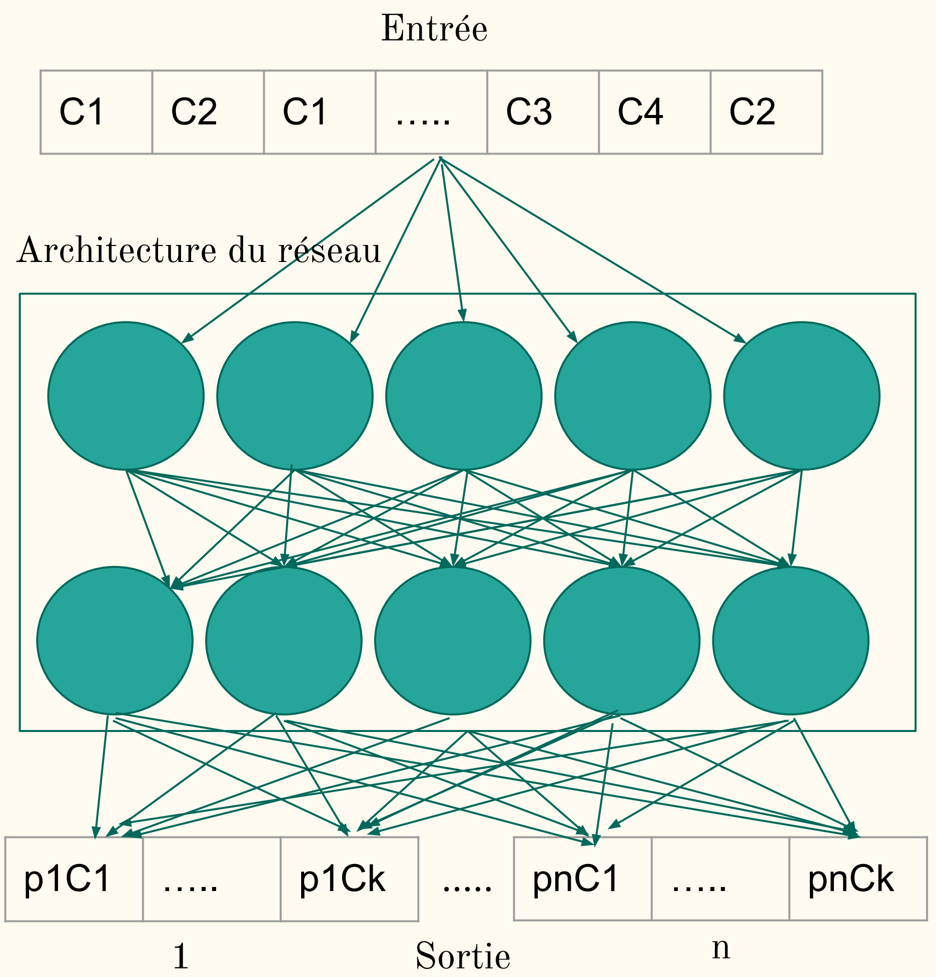
Nombres de Schur



- Entrée: Vecteur des couleurs du tableau
- Ensemble de couleurs C={Ci}
- Architecture de réseau à chercher
- Sortie: Probabilités piCj que la position i du tableau soit de la couleur Cj



Fonction de côut à chercher, exemple possible:

$$f(Sortie) = \frac{1}{1 + \sum_{i=1}^{k} \sum_{1 \leq j,k,l \leq n} (\theta(p_j c_i) + \theta(p_k c_i) - \theta(p_l c_i))^2}$$

Utilisation d'un algorithme génétique pour l'amélioration des poids des neurones.

Out-of-order

```
Data: n ≥ 0
Result: BestSolution
Solutions ← Dict(keys : {1 ... n});
num ← 1;
while BreakCondition(n,num, Choice) do
    num ← num + 1;
    for sol_i ∈ Solutions do
        sol_i.append(num);
        Fitness[i] ← EvaluateFitness(sol_i);
    end
    bestSolution ←
     Choice(Solutions, Fitness);
end
```

Lemma 1. *Let* $S = \bigsqcup_i S_i$ *be a weakly sum-free partition such that its size* $|S| = \sum_i |S_i| = n$, *and* $fitness(S) = k$. *Then, assume we add* $N \in \mathbb{N} - S$ *to* S *i.e.* $\exists i \in [n] : S_i \leftarrow S_i \cup \{N\}$. *Then,* $fitness(S \cup \{N\}) \geq k$.

And in general,

$$0 \leq fitness(S \cup \{N\}) - fitness(S) \leq g(k_i) = O(k_i^2)$$

Data: $n \geq 0$, $m \geq 1$
Result: $BestSolution$
$Solutions \leftarrow Dict(keys : \{1 \ldots m\}, values :$
$Dict(keys : \{1 \ldots n\})\ );$
while *BreakCondition(n,num, Choice)* **do**
    for $Process_j$ *in* $ProcessPool$ **do**
        $num \leftarrow getNumber(num, j);$
        for $sol_i \in Solutions[j]$ **do**
            $sol_i.\mathrm{append}(num);$
            $Fitness[j][i] \leftarrow$
            $EvaluateFitness(sol_i);$
        end
        $BestSolutions[j] \leftarrow$
        $Choice(Solutions[j], Fitness[j]);$
    end
    $BestSolution \leftarrow$
    $Choice(BestSolutions, Fitness);$
end