Automatisation de la conception d'horaire de professeurs

Les nageuses synchronisées de Pierrefonds
Équipe #3

Pierre-Olivier Blouin - BLOP11068701
Samuel Lambert - LAMS05028203
Richard Pigeon - PIGR28059108
Marc-André Poulette - POUM24058905

Introduction

On s’intéresse à la tâche de générer des associations cours-professeurs.

Problématique intéressante, car cela nous a permis de mettre en pratique des techniques autres que celles utilisées dans les travaux pratiques.

Est-ce possible d’automatiser cette tâche?

Problématique à résoudre

- Opération qui est souvent effectuée «à la main».

- Chaque professeur présente une liste des cours qu'il souhaite enseigner (par ordre de priorité) ainsi que le nombre de cours qu'il aimerait donner à la prochaine session.

- Problème à satisfaction de contraintes.

Problématique à résoudre

Contraintes

- Un groupe--cours peut être assigné seulement une fois.
- Un professeur ne peut pas donner deux cours durant la même plage horaire.
- Un professeur ayant une mauvaise évaluation pour un cours ne peut plus donner ce
- cours.
- Il y a un seul directeur et celui--ci a priorité sur tout.
- Un professeur a priorité sur un chargé de cours.
- Un directeur ne peut donner plus d’un cours.
- Un professeur ne peut donner plus de deux cours.
- Un chargé de cours ne peut donner plus de quatre cours.

Techniques de résolution

Comment on peut résoudre ce problème?

- Recherche «naïve»
- Recherche locale
- *Backtracking Search*
- Autre ?

Techniques de résolution

Méthodes de résolution de prédilection :

- *Backtracking Search*
- *AC-3*

Ces techniques permettent de résoudre le problème de façon «naturelle».

Rappels - *CSP*

Un problème de satisfaction de contraintes est défini par :

- Ensemble fini de variables X_1, …, X_n
- Chaque variable X_i a un domaine D_i de N valeurs possibles
- Ensemble fini de contraintes C_1, …, C_m sur les variables

Rappels - *CSP*

Dans le cas d'un générateur d'horaire

Variables : Les professeurs à qui on doit assigner des cours
ex: Variables = {prof1, prof2, prof3}

Domaines : Les cours (en ordre de préférence) que les professeurs veulent enseigner. Cette liste peut être de longueur variable.
ex: Domaine(prof1) = {INF1120, INF2120, INF3105}

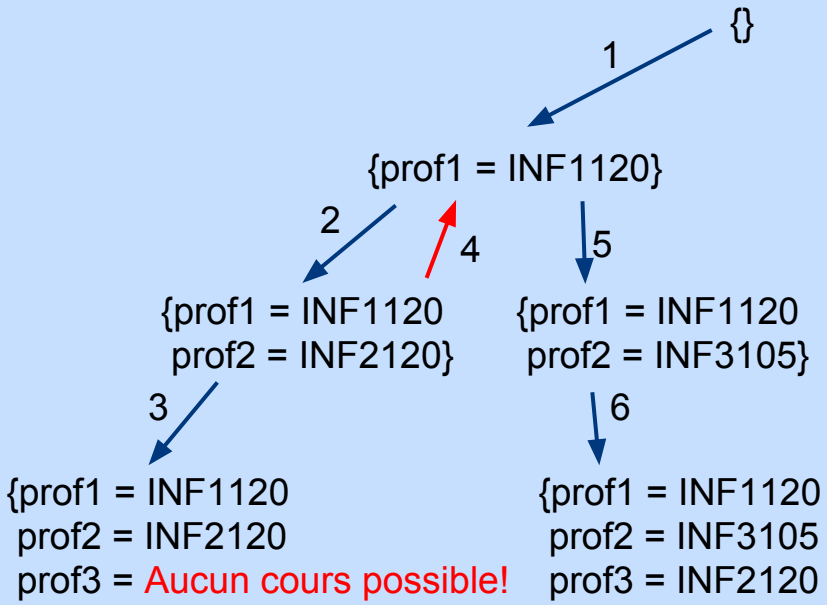
Contraintes : On restreint les valeurs pour un sous-ensemble de variables
ex: C_1 = Un professeur ne peut pas donner plus de 2 cours

Rappels - *Backtracking Search*

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences ← INFERENCE(csp, var, value)
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACKTRACK(assignment, csp)
                if result ≠ failure then
                    return result
        remove {var = value} and inferences from assignment
    return failure
```

Rappels - *Backtracking Search*



Notre but :

Obtenir une assignation complète et consistante.

Dans le cadre de notre générateur d'horaire, nous voulons que chaque professeur puisse enseigner le nombre de cours qu'il désire. De plus, ces cours doivent être sélectionnés parmi leur liste de cours désirés.

Rappels - *AC-3*

function AC-3(csp) **returns** false if an inconsistency is found and true otherwise
  inputs: csp, a binary CSP with components $(X,\ D,\ C)$
  local variables: $queue$, a queue of arcs, initially all the arcs in csp

  while $queue$ is not empty **do**
    $(X_i,\ X_j) \leftarrow$ REMOVE-FIRST($queue$)
    if REVISE(csp, $X_i,\ X_j$) **then**
      if size of $D_i\ =\ 0$ **then return** $false$
      for each X_k **in** X_i.NEIGHBORS - $\{X_j\}$ **do**
        add $(X_k,\ X_i)$ to $queue$
  return $true$

---

function REVISE(csp, $X_i,\ X_j$) **returns** true iff we revise the domain of X_i
  $revised \leftarrow false$
  for each x **in** D_i **do**
    if no value y in D_j allows (x,y) to satisfy the constraint between X_i and X_j **then**
      delete x from D_i
      $revised \leftarrow true$
  return $revised$

Rappels - *AC-3*

Dans le cas d'un générateur d'horaire

Imaginez qu'on a le problème suivant :
Professeurs = {prof1, prof2 prof3}

Chaque professeur veut donner un seul cours parmi leurs choix (classés en ordre de priorité).

Choix :
prof1 = {INF1120, INF2120, INF3105}
prof2 = {INF1120, INF2120}
prof3 = {INF1120}

Inéfficace ! Le nombre de *backtracks* sera trop grand !

Rappels - *AC-3*

Dans le cas d'un générateur d'horaire

L'exécution de l'algorithme *AC-3* avant l'appel du *Backtracking Search* va modifier les choix de chacun des professeurs à ceci :

Choix :
prof1 = {INF3105}
prof2 = {INF2120}
prof3 = {INF1120}

Évidemment, le problème sera beaucoup plus facile à resoudre!

Rappels - *Complexité temporelle/spatiale*

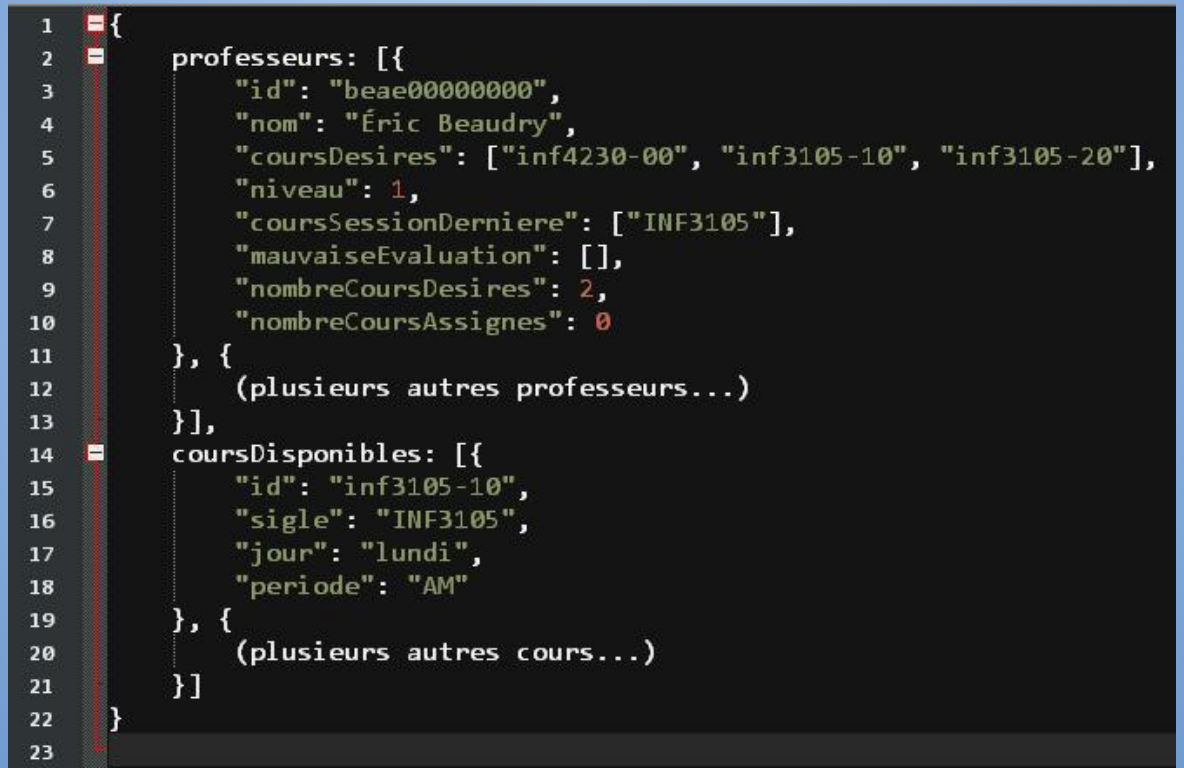
Soient *c* le nombre d'arcs, *d* la taille du plus grand domaine d'une variable et *n* le nombre de variables :

Backtracking Search
Complexité temporelle : $O(d^n)$ dans le pire cas
Comlpexité spatiale : $O(dn)$

AC-3
Complexité temporelle : $O(c^2d^3)$ dans le pire cas.
Complexité spaciale : $O(c)$.

Détails sur l'implémentation

Exemple d'un objet CSP



```
{
    professeurs: [{
        "id": "beae00000000",
        "nom": "Éric Beaudry",
        "coursDesires": ["inf4230-00", "inf3105-10", "inf3105-20"],
        "niveau": 1,
        "coursSessionDerniere": ["INF3105"],
        "mauvaiseEvaluation": [],
        "nombreCoursDesires": 2,
        "nombreCoursAssignes": 0
    }, {
        (plusieurs autres professeurs...)
    }],
    coursDisponibles: [{
        "id": "inf3105-10",
        "sigle": "INF3105",
        "jour": "lundi",
        "periode": "AM"
    }, {
        (plusieurs autres cours...)
    }]
}
```

Détails sur l'implémentation

Code de la fonction *backtrackingSearch*

```
70  function backtrackingSearch(csp, assignment) {
71      if (isComplete(assignment)) return assignment;
72
73      var professeur = selectNextUnassignedVariable(csp, PROFESSEUR);
74      if (!professeur) professeur = selectNextUnassignedVariable(csp, CHARGE_DE_COURS);
75
76      var domaineProfesseur = orderDomainValues(professeur, assignment, csp);
77      var result;
78
79      for (var i = 0; i < domaineProfesseur.length; i++) {
80          var cours = getCoursById(csp, domaineProfesseur[i]);
81          var assignmentCopy = JSON.parse(JSON.stringify(assignment));
82
83          addAssignment(professeur, cours["id"], assignment);
84
85          if (isConsistent(cours, professeur, assignmentCopy)) {
86              var result = backtrackingSearch(csp, assignment);
87              if (result) break;
88          }
89
90          removeAssignment(professeur, cours, assignment);
91      }
92      return result;
93  }
```

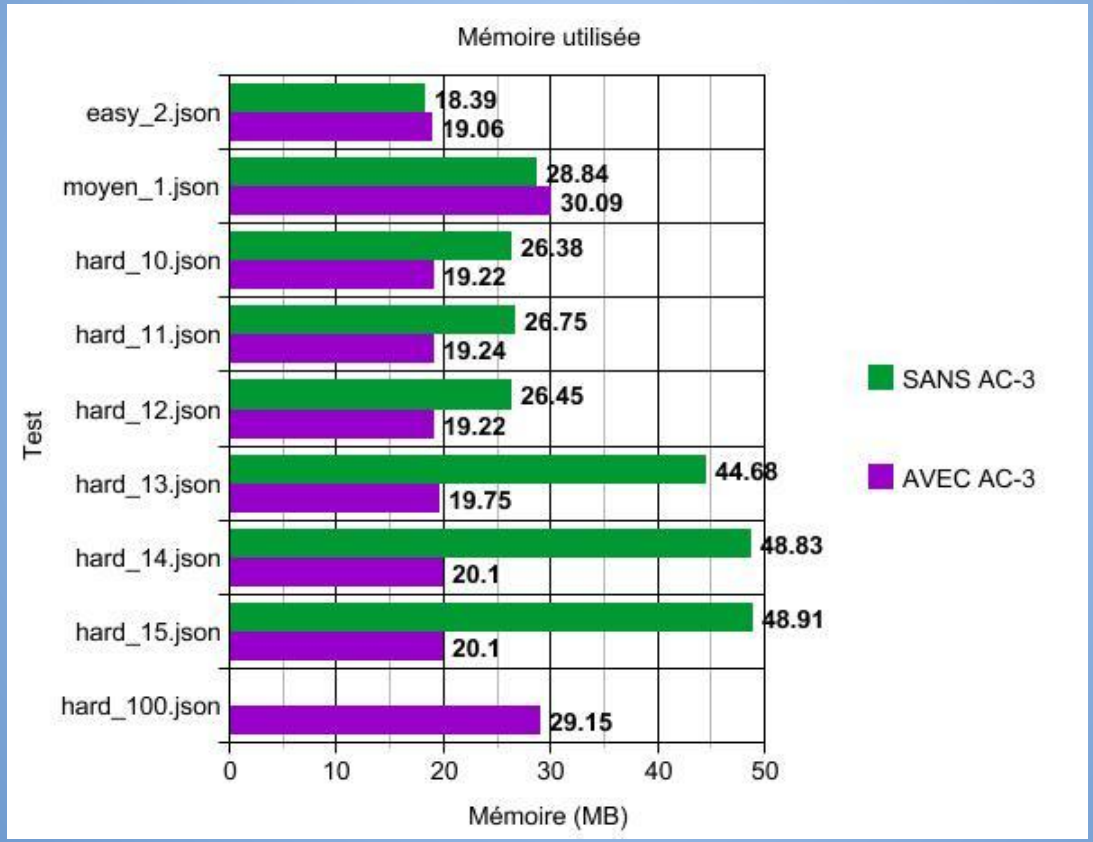
Démonstration...

Résultats

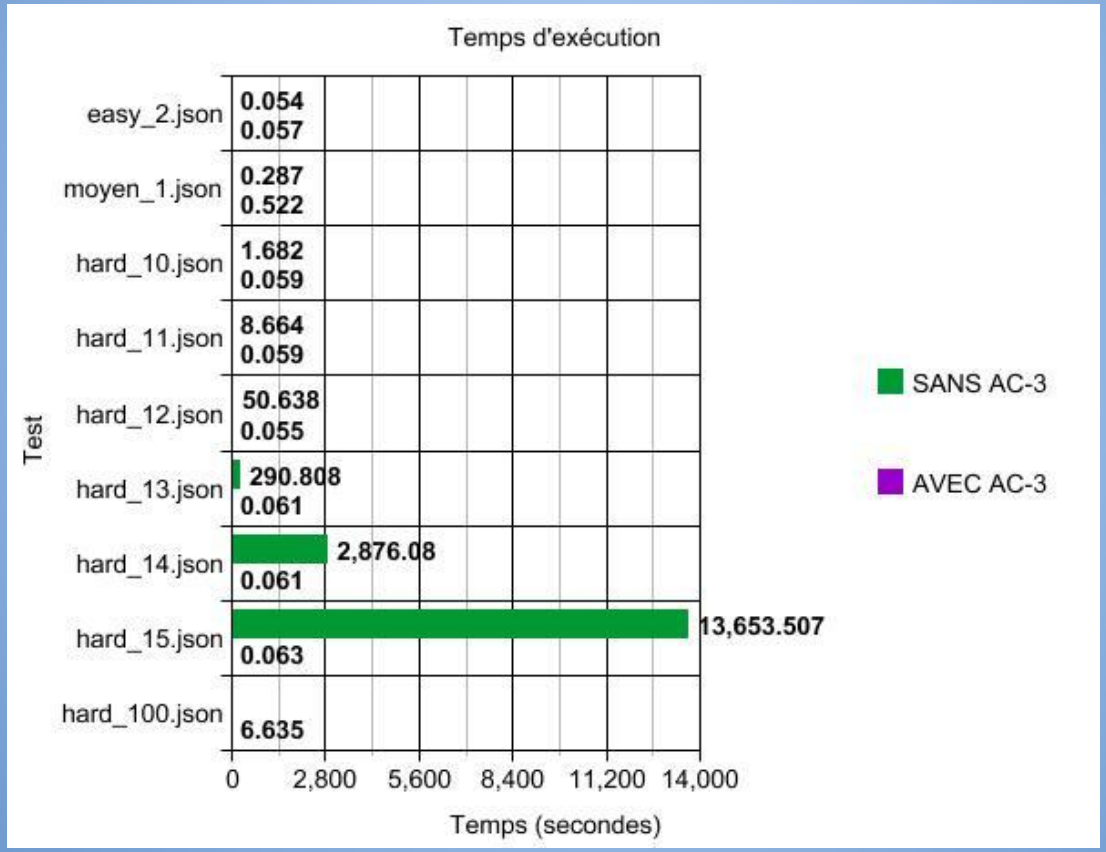
Il s'est avéré difficile de générer des problèmes réalistes, difficiles et valides afin d'effectuer nos tests.
Voici une courte description des jeux de données utilisés pour les tests :

Fichier de test	Nombre de professeurs	Nombre de cours
csp_easy_2.json	6	6
csp_moyen_1.json	171	1000
csp_hard_10.json	10	10
csp_hard_11.json	11	11
csp_hard_12.json	12	12
csp_hard_13.json	13	13
csp_hard_14.json	14	14
csp_hard_15.json	15	15
csp_hard_100.json	100	100

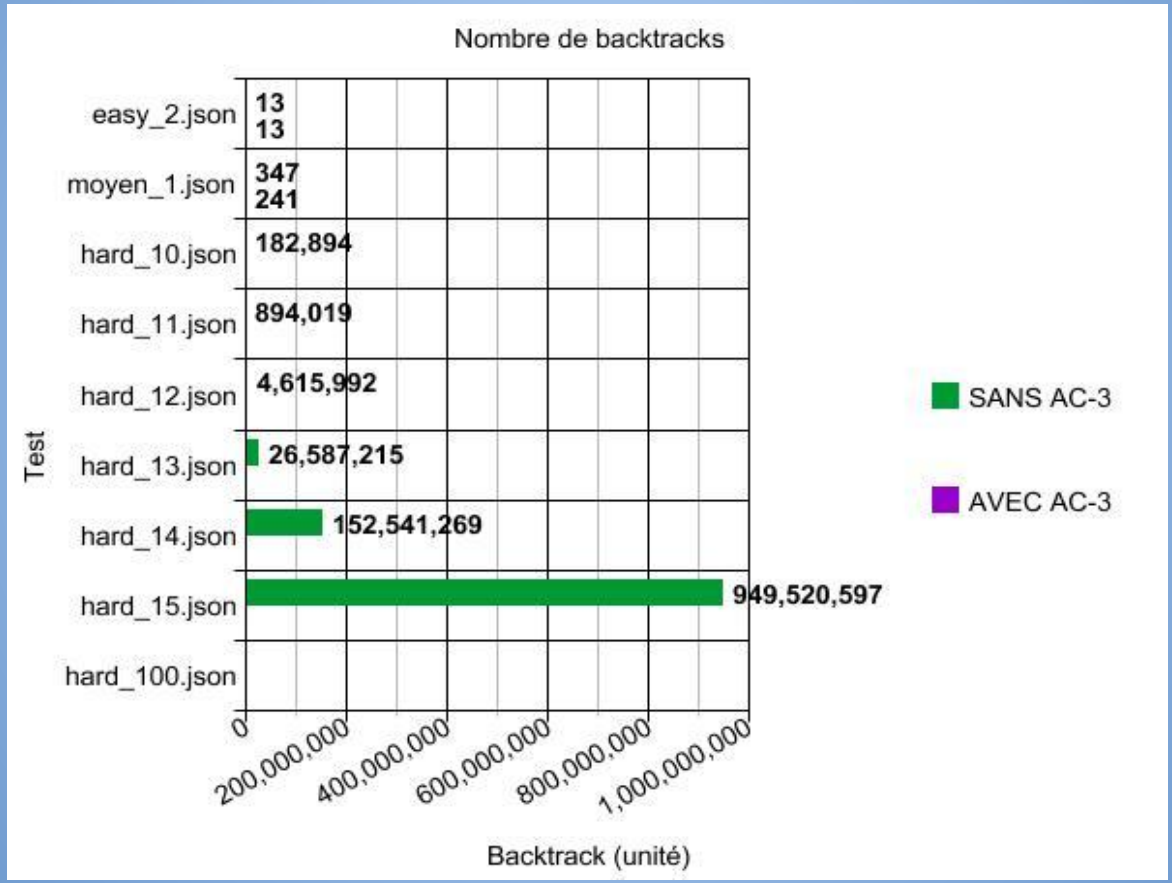
Résultats (Mémoire)



Résultats (Temps)



Résultats (Backtracks)



Conclusion

- Algorithme *CSP* répond à la problématique
- Très efficace avec l'utilisation du *AC-3*.
- Pistes pour amélioration?
  - Il serait intéressant d'avoir des contraintes plus réalistes.
  - Il serait intéressant d'implémenter la notion d'historique.
  - Mettre notre programme en pratique avec de vraies données, celles de l'UQÀM, par exemple. Sinon, on pourrait créer un générateur de données plus réaliste.
  - Gérer les erreurs de façon plus efficace. Si un problème est *impossible*, on devrait quand même assigner le plus de cours plutôt que de ne rien retourner.

Conclusion

Des questions ?