Master 2 Informatique Fondamentale et Appliquée Année 2024-2025

Carole Delporte, Amélie Gheerbrant, Giulio Manzonetto, Ralf Treinen

2/12/2024

Calendrier 24/25

- Deuxième période : du 06/01 au 21/03 (cours/TD/TP)
  Examens : du 24/03 au 28/03
- Troisième période : du 31/03 au 30/09 (stage)
  Soutenance de stage, et jury du M2 : mi-septembre

Rappel : Vous devrez valider 60 crédits

- 11 modules d'informatique (chacun à 3 crédits), modules à choisir selon les règles spécifiques de votre parcours.
- le module d'Anglais, première période (3 crédits)
- le stage en entreprise pendant la troisième période (24 crédits)

Les règles du parcours DATA

Anglais + 11 cours d'Info, dont au moins 7 *cours fondamentaux* :

1ère période	2ème période
Architecture des Systèmes de Bases de Données	Architecture des Systèmes d'Information
Base de données spécialisées	Algorithmique Répartie
Fouille de Données et Aide à la Décision	Introduction Apprentissage Profond
Programmation Objets : Concepts Avancées	Programmation Logique et Par Contraintes Avancée
	Grands Réseaux d'Interaction
	Optimisation
	Programmation Répartie

Les règles du parcours IMPAIRS

Anglais + 11 cours d'Info, dont au moins 8 *cours suggérés* :

1ère période	2ème période
Architecture des Systèmes de BD	Architecture des SI
Fouille de données et aide à la décision	Interfaces et Outils de MacOS-X
Informatique embarquée	Administration système et réseaux
Ingénierie des protocoles	Algorithmique répartie
Modélisation et Spécification	Programmation répartie
Programmation Synchrone	Grands réseaux d'interaction
Protocoles des services Internet	Mobilité
	Systèmes Avancés

Les règles du parcours LP

Anglais + 11 cours d'Info, dont 2 *cours obligatoires* :

1ère période	2ème période
Programmation Objets : Concepts Avancées	Programmation Comparée

et au moins 6 parmi les *cours recommandés* :

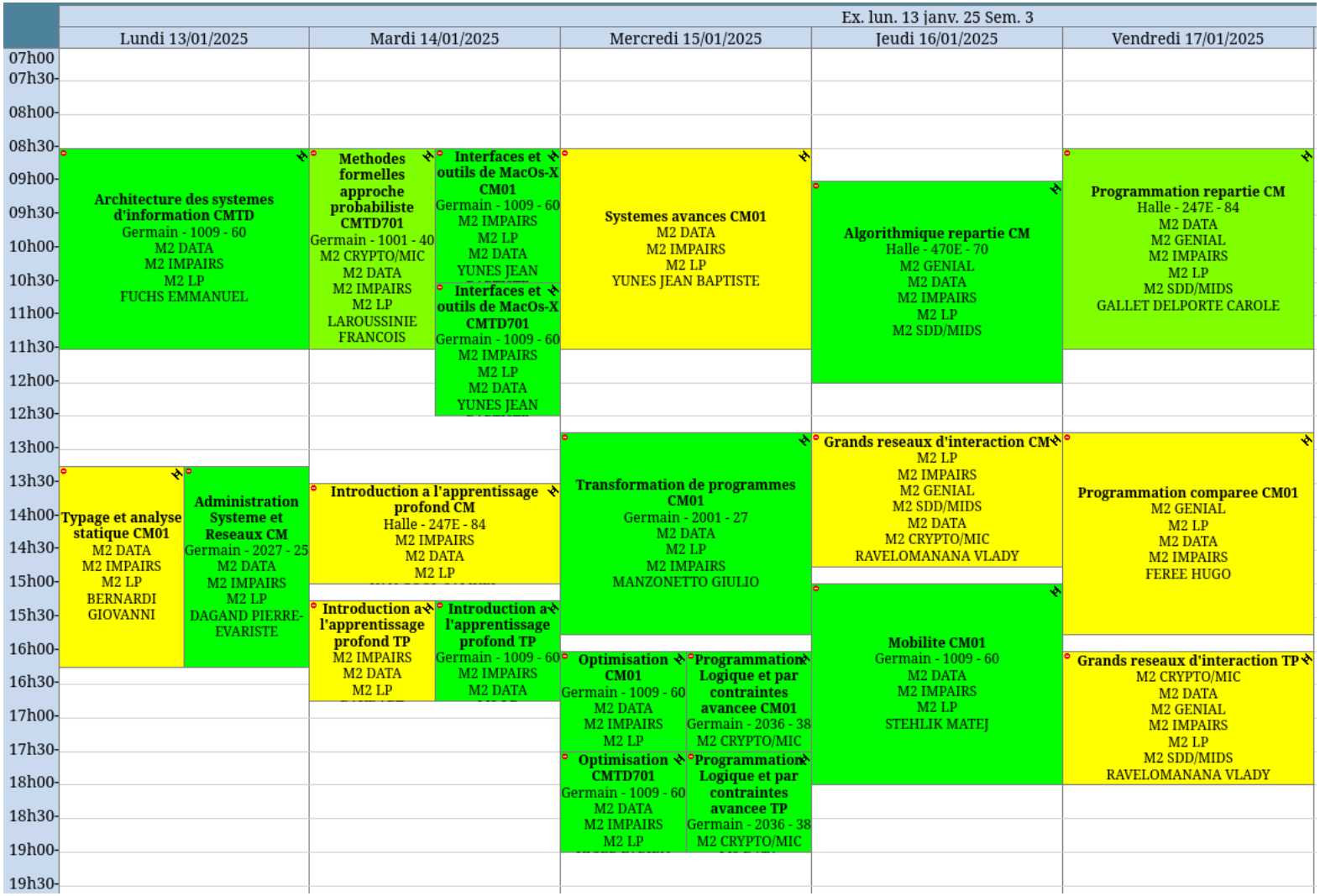
1ère période	2ème période
Architecture des Systèmes de BD	Architecture des SI
Méthodes Formelles de Vérification	Programmation Logique et Par Contraintes Avancée
Modélisation et Spécification	Programmation Répartie
Programmation Synchrone	Analyse Statique
	Interfaces et Outils de MacOS-X
	Transformation de Programmes

Inscriptions pédagogiques à des modules

- Vous choisissez sur silice les cours que vous voulez suivre.
- Sauf les redoublants : email au responsable de parcours.
- Vous choisissez 6 cours pour la deuxième période.
- Si vous souhaitez un cours en plus : l'indiquer également sur silice. Ces demandes seront traitées après les demandes des 6 cours. On va essayer de vous y inscrire (sans promesse).
- Dans les trois parcours on peut aussi choisir un cours "externe" (par ex au MPRI, LMFI, une autre université) *avec accord du responsable de parcours*.

Inscriptions pédagogiques à des modules

- Ouverture des IP pour la deuxième période : du 2/12 au 8/12.
- Vous ne pouvez pas vous inscrire à des cours qui ont lieu au même moment :
  - *Administration Systèmes Réseaux* - *Analyse Statique*
  - *Méthodes Formelles Proba* - *Interfaces & Outils MacOSX*
  - *Optimisation* - *Prog Logique & Contraintes Avancée*
- Les cours ont des capacités limitées. La priorité sera donnée aux étudiants pour qui le cours demandé est recommandé.



Ex. lun. 13 janv. 25 Sem. 3

	Lundi 13/01/2025	Mardi 14/01/2025	Mercredi 15/01/2025	Jeudi 16/01/2025	Vendredi 17/01/2025
07h00					
07h30					
08h00					
08h30					
09h00	**Architecture des systemes d'information CMTD** Germain - 1009 - 60 M2 DATA M2 IMPAIRS M2 LP FUCHS EMMANUEL	**Methodes formelles approche probabiliste CMTD701** Germain - 1001 - 40 M2 CRYPTO/MIC M2 DATA M2 IMPAIRS M2 LP LAROUSSINIE FRANCOIS **Interfaces et outils de MacOs-X CM01** Germain - 1009 - 60 M2 IMPAIRS M2 LP M2 DATA YUNES JEAN **Interfaces et outils de MacOs-X CMTD701** Germain - 1009 - 60 M2 IMPAIRS M2 LP M2 DATA YUNES JEAN	**Systemes avances CM01** M2 DATA M2 IMPAIRS M2 LP YUNES JEAN BAPTISTE		**Programmation repartie CM** Halle - 247E - 84 M2 DATA M2 GENIAL M2 IMPAIRS M2 LP M2 SDD/MIDS GALLET DELPORTE CAROLE
09h30				**Algorithmique repartie CM** Halle - 470E - 70 M2 GENIAL M2 DATA M2 IMPAIRS M2 LP M2 SDD/MIDS	
10h00					
10h30					
11h00					
11h30					
12h00					
12h30					
13h00			**Transformation de programmes CM01** Germain - 2001 - 27 M2 DATA M2 LP M2 IMPAIRS MANZONETTO GIULIO	**Grands reseaux d'interaction CM** M2 LP M2 IMPAIRS M2 GENIAL M2 SDD/MIDS M2 DATA M2 CRYPTO/MIC RAVELOMANANA VLADY	**Programmation comparee CM01** M2 GENIAL M2 LP M2 DATA M2 IMPAIRS FEREE HUGO
13h30	**Typage et analyse statique CM01** M2 DATA M2 IMPAIRS M2 LP BERNARDI GIOVANNI **Administration Systeme et Reseaux CM** Germain - 2027 - 25 M2 DATA M2 IMPAIRS M2 LP DAGAND PIERRE-EVARISTE				
14h00		**Introduction a l'apprentissage profond CM** Halle - 247E - 84 M2 IMPAIRS M2 DATA M2 LP			
14h30					
15h00					
15h30		**Introduction a l'apprentissage profond TP** M2 IMPAIRS M2 DATA M2 LP **Introduction a l'apprentissage profond TP** Germain - 1009 - 60 M2 IMPAIRS M2 DATA		**Mobilite CM01** Germain - 1009 - 60 M2 DATA M2 IMPAIRS M2 LP STEHLIK MATEJ	
16h00			**Optimisation CM01** Germain - 1009 - 60 M2 DATA M2 IMPAIRS M2 LP **Programmation Logique et par contraintes avancee CM01** Germain - 2036 - 38 M2 CRYPTO/MIC		**Grands reseaux d'interaction TP** M2 CRYPTO/MIC M2 DATA M2 GENIAL M2 IMPAIRS M2 LP M2 SDD/MIDS RAVELOMANANA VLADY
16h30					
17h00					
17h30					
18h00			**Optimisation CMTD701** Germain - 1009 - 60 M2 DATA M2 IMPAIRS M2 LP **Programmation Logique et par contraintes avancee TP** Germain - 2036 - 38 M2 CRYPTO/MIC		
18h30					
19h00					
19h30					

Programmation comparée – 2025

Hugo Férée, Julien Narboux
{feree,narboux}@irif.fr

2025

Objectifs du module

Enrichir votre culture technique

- **Apercevoir** différents concepts des langages de programmation.
  - En largeur plutôt qu'en profondeur.
- **Comparer** des langages, environnements ou principes de programmation.
- Apprendre à développer un **avis critique** argumenté sur un sujet technique.

Améliorer votre expression orale et écrite

- Préparer et donner des **exposés** clairs sur des sujets techniques précis.
  - En profondeur cette fois-ci.
- Écrire des **rapports** synthétiques.
  - Clairs et concis.

Prérequis du module

- Un **intérêt** pour la programmation et sa pratique.
- Une familiarité avec plusieurs **paradigmes** de programmation: **impératif, fonctionnel, objet**.

Une séance typique

Format standard : comparaison technique de technologies voisines ou d'un même concept dans plusieurs langages.

(**Alternative** : *focus* sur une seule technologie, analyse technique/conceptuelle étoffée.)

Déroulement :

- Des exposés (binômes ; 20+10 minutes ; doit contenir du code original)
- Une discussion globale après chaque exposé, sur le fond et sur la forme.

Travail à réaliser

- 2 exposés (en binômes) sur des sujets différents
- 2 rapports (individuels) sur les exposés des autres groupes

Exemples de sujets

- POO : nominal (Java) vs. structurel (OCaml) vs. prototype (JavaScript)
- Effets dans les langages fonctionnels : OCaml vs. Haskell vs. Koka
- Programmation modulaire : modules (ML/OCaml) vs. traits (Scala)
- Gestion mémoire : manuelle (p. ex. C), GC traçant (Java, OCaml...), basée sur RAII/borrowing (Rust)
- Extensibilité linguistique : CPP vs. décorateurs Python vs. OCaml PPX
- Focus : Rust dans le noyau Linux
- Language Integrated Query: LINQ
- Programmer avec des continuations: des langages fonctionels aux frameworks pour le web

Méthodes formelles
Approches probabilistes

François Laroussinie

2024-2025

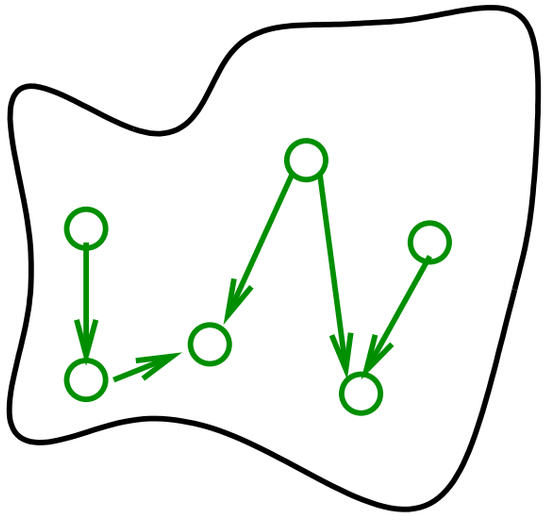
Mardi 8h30-11h30, (à partir du 14 janvier !!!)

Model-checking

Vérification automatique

System ***Properties***

Formalizing step



$\models$?

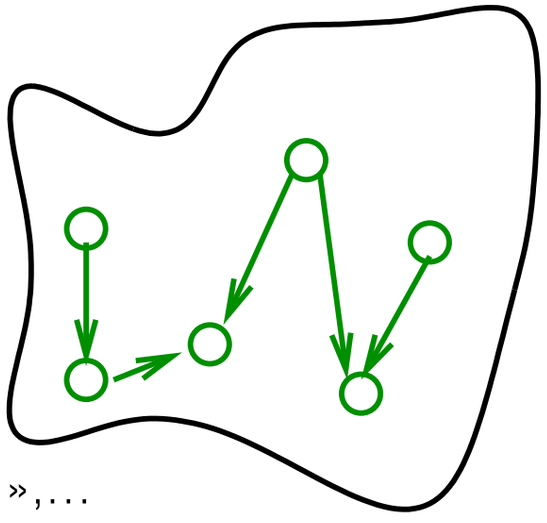
φ

Model-checking

Vérification automatique

System ***Properties***

Formalizing step



$\models$?

φ

Graphe étiqueté,
réseau de Petri,
automates à « bidules »,…

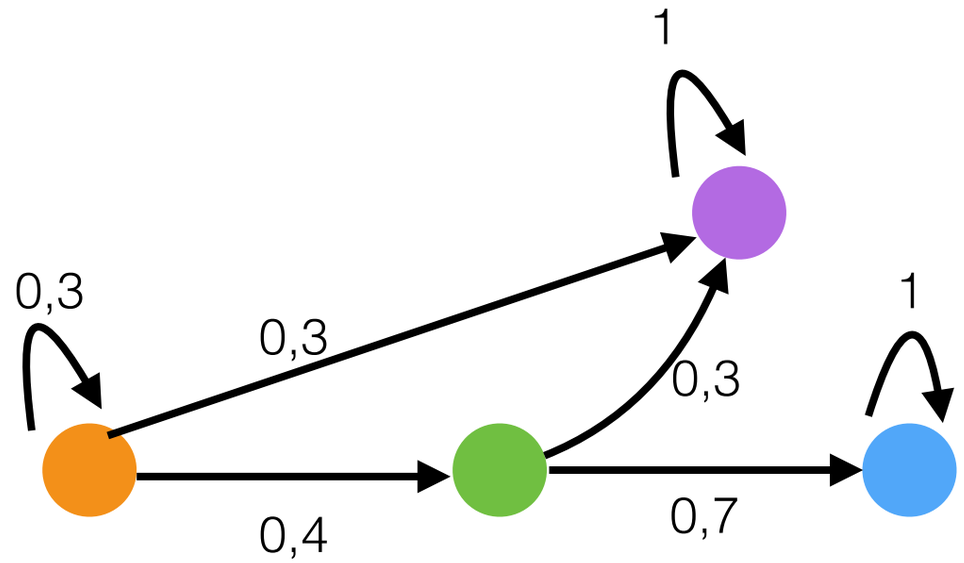
Logiques temporelles:
LTL, CTL,…

Modèles:

- chaînes de Markov
- processus de décision de Markov

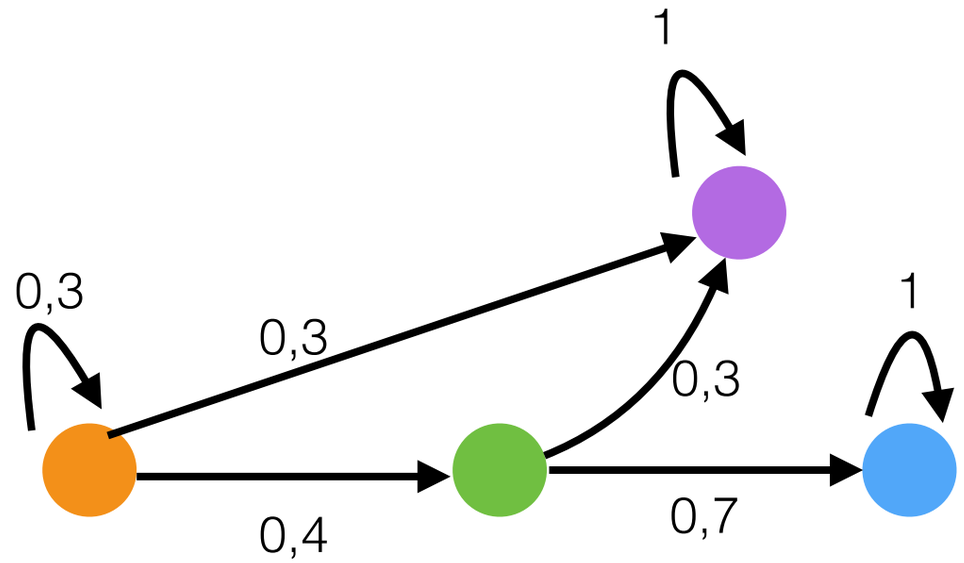
Modèles:

- chaînes de Markov
- processus de décision de Markov



Modèles:

- chaînes de Markov
- processus de décision de Markov

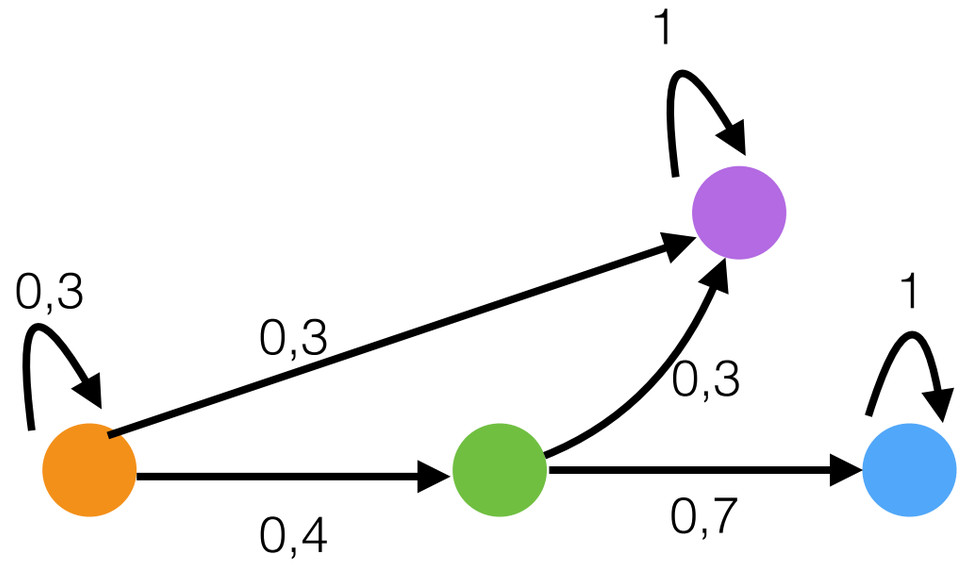


Propriétés:

- Accessibilité
- Logique temporelle probabiliste (PCTL)

Modèles:

- chaînes de Markov
- processus de décision de Markov



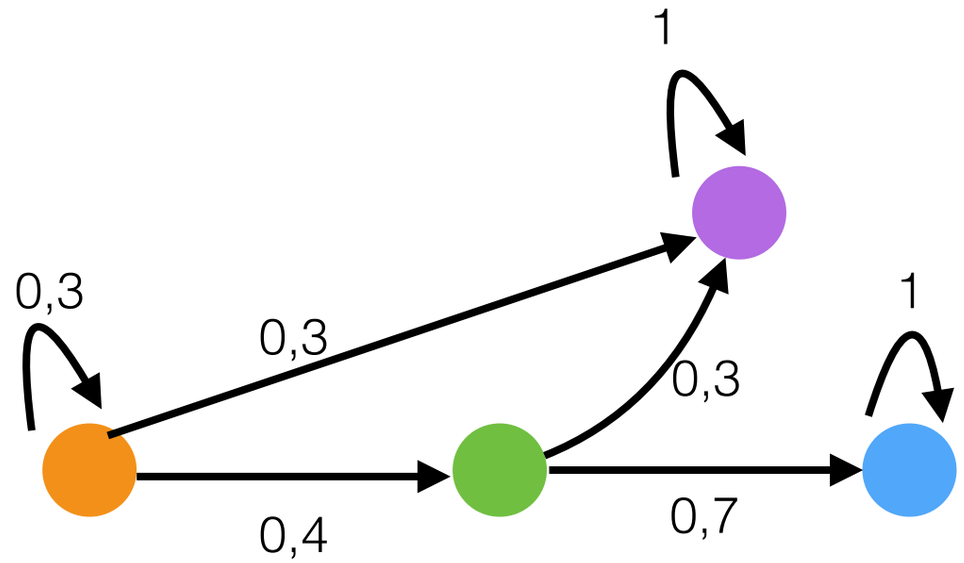
Propriétés:

- Accessibilité
- Logique temporelle probabiliste (PCTL)

● ⊨ $P_{>0.5}$(F ●) ?

Modèles:

- chaînes de Markov
- processus de décision de Markov



Propriétés:

- Accessibilité
- Logique temporelle probabiliste (PCTL)

● $\models P_{>0.5}(F$ ● $)$?

Important:

- Apprécier l'algorithmique (dans les graphes) et la logique (temporelle).
- Pas d'allergie aux probabilités.

Méthodes formelles
Approches probabilistes

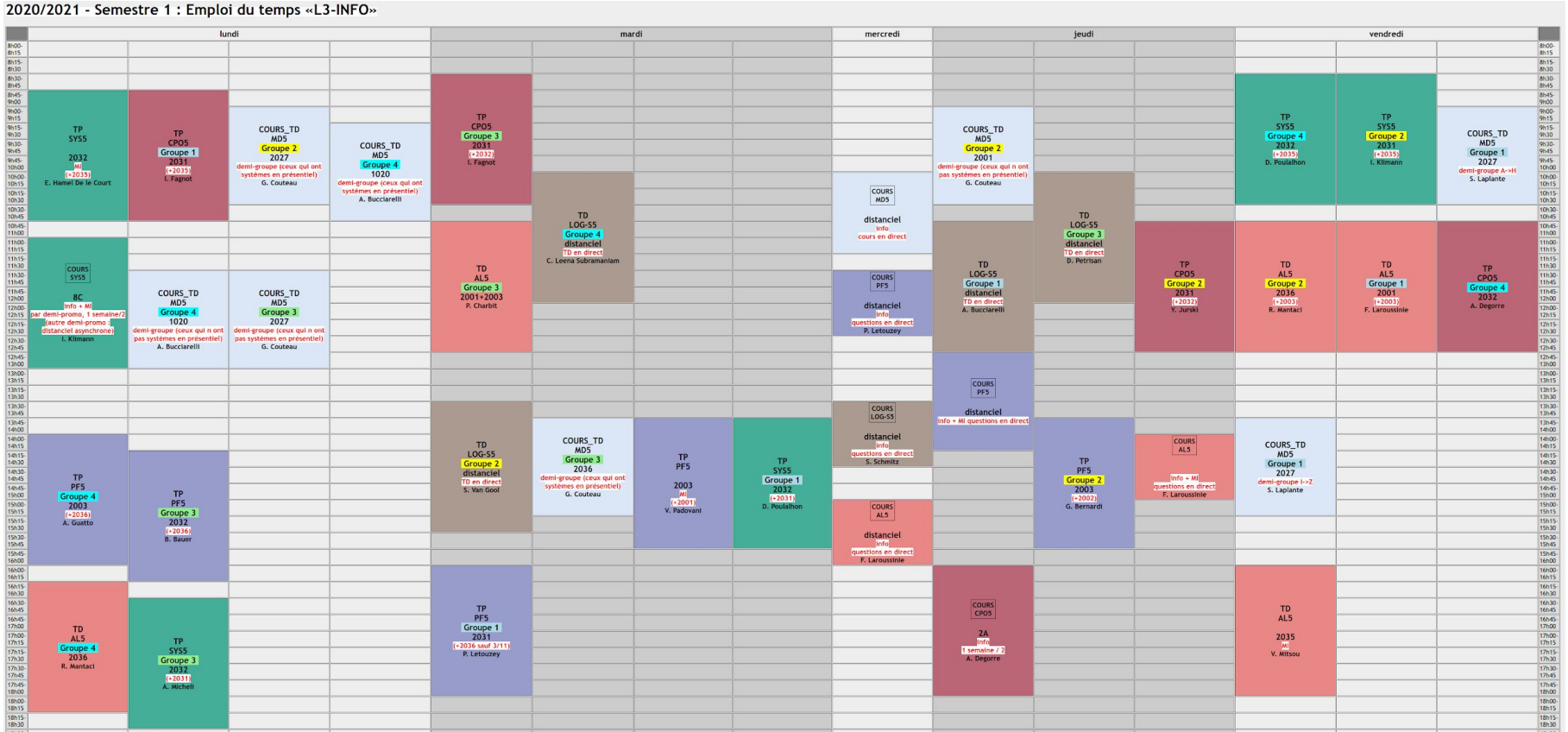
- Modéliser et spécifier des systèmes probabilistes
- Comprendre les algorithmes
- Utiliser des outils (prism)

Optimisation Combinatoire

http://fabien.viger.free.fr/oc

Qu’est-ce que l’Optimisation Combinatoire ?

Exemple: Création des emploi du temps



Qu’est-ce que l’Optimisation Combinatoire ?

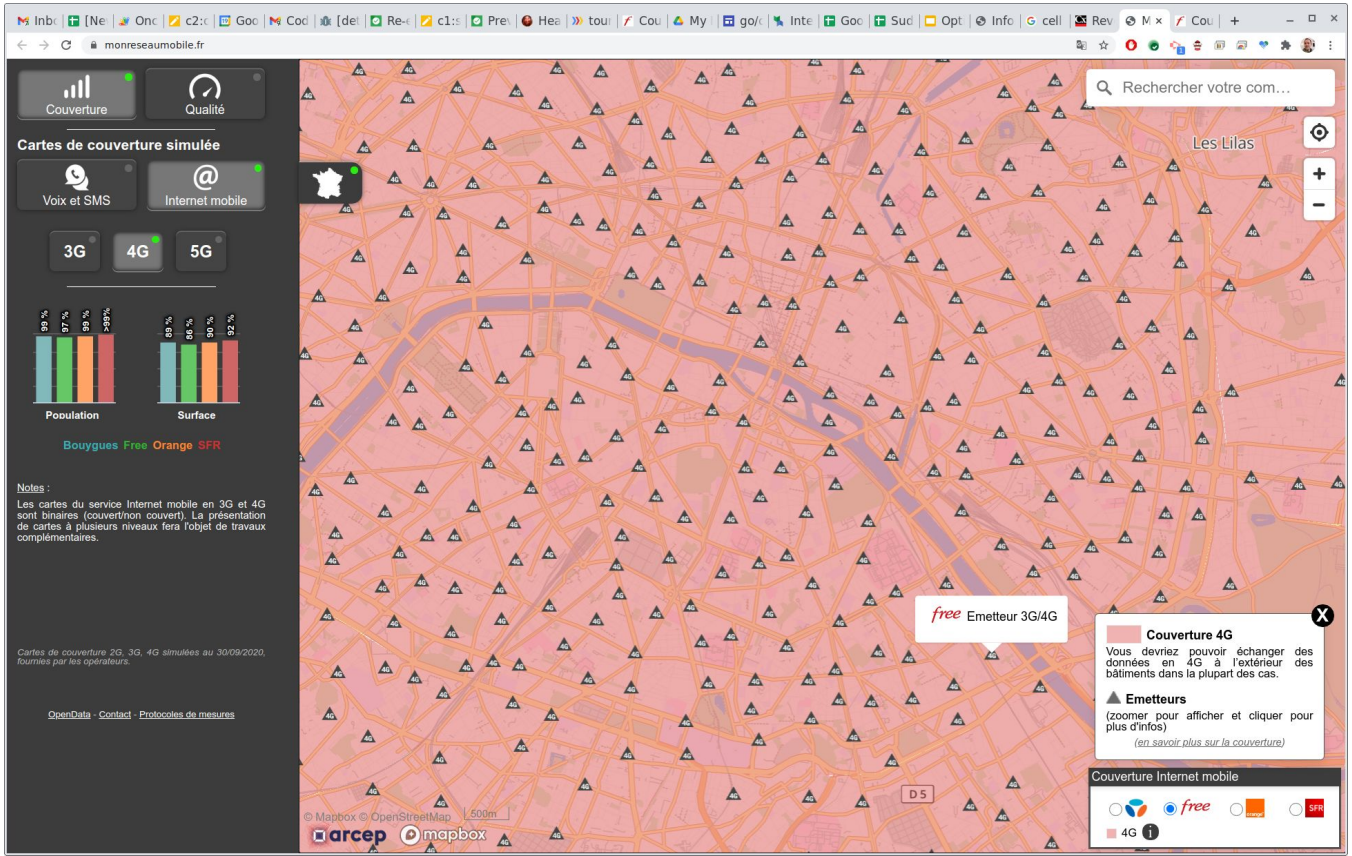
Autre exemple: Sudoku (cliquer sur l’image)

Input Grid

	6			5			2	
							9	
7							1	
		6		3		4		
		4		7		1		
		5		9		8		
	4				1			6
	3				8			
	2			4			5	

Qu’est-ce que l’Optimisation Combinatoire ?

Autre exemple: où placer les émetteurs 4G ?



Organisation du Cours

Mercredi 15h-18h30, Sophie Germain: 1h30 cours + 1h45 TD

Note = **50% Examen Final** (sur papier) et **50% TD**

Cours au tableau : pas de support de cours, mais vidéos dispo après chaque cours pour réviser si besoin.

TD en C++, avec **Auto-Tests;** corrigés dispos après le TD.

- Adaptés aux débutants en C++ (mais aux experts aussi)
- TD = application *directe* du cours.

Plan indicatif du Cours

1. **Graphes**. Implémentations (C++). Composantes [fortement] connexes. Tables de hachage. Parcours (BFS, DFS). Dijkstra.
2. **Arbres** et leurs algorithmes. Autres graphes particuliers (DAG, **Tri topologique**). Programmation dynamique: 1er exemple.
3. **Programmation dynamique**: autres exemples.
4. **Algorithmes Gloutons:** Arbre couvrant minimum et autres.
5. **Heuristiques, Monte-Carlo** (exemples: diamètre d'un graphe. Miller-Rabin).
6. **Programmation linéaire** (*aka* LP)
7. Programmation linéaire **entière** (*aka* **MIP**)
8. MIP: Modélisation avancée.
9. **Examen blanc** commenté

Extensions possible du Cours

- **SAT** (CDCL)
- **Max-Flow**, Min-cost flow
- Revisions d'algorithmique: Hash tables. Tas. Complexité.

Tout est dispo sur :

http://fabien.viger.free.fr/oc

Architecture des Systèmes d'Information

Emmanuel Fuchs

INFORMATIQUE
Sciences
Université Paris Cité

Architecture des Systèmes d'Information

- Bibliographie des systèmes d'information.
- Historique des Technologies de l'Information (IT) et Opérationnelle (OT).
- Architecture d'Entreprise
- Architecture Logicielle et Matérielle des systèmes d'information
- Cloud Computing et Data Center
- Framework d'Entreprise
- Framework Logiciel
- .....

INFORMATIQUE
Sciences
Université Paris Cité

Administration Système et Réseaux

Pierre-Évariste Dagand
doc@evr.ist

December 2nd, 2024

Administration Système et Réseaux

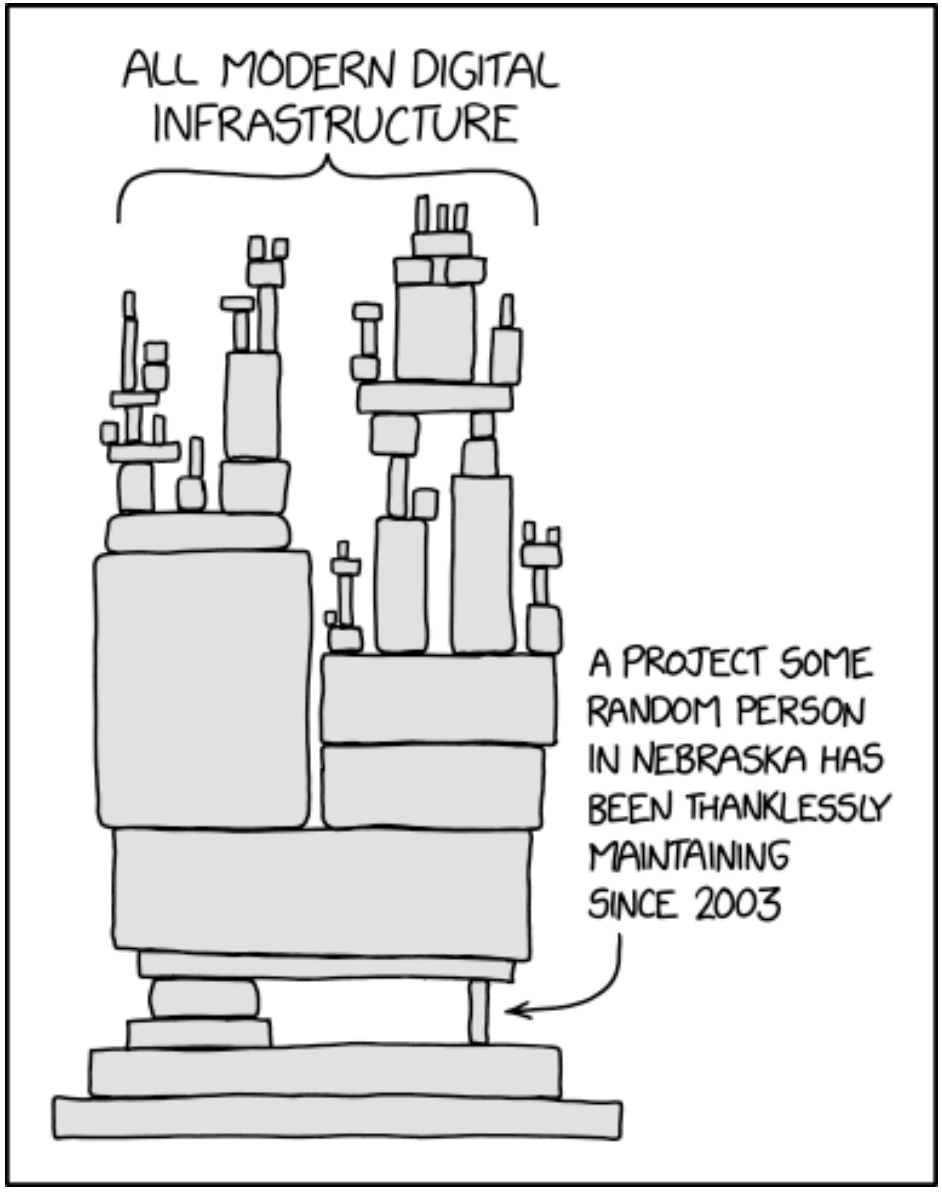


Figure: `xkcd` #2347

"Any sufficiently advanced technology is indistinguishable from magic."

Arthur C. Clarke

Administration Système et Réseaux

Objectives

- Deploy networked software
- Gain observability throughout the software stack
- Understand live software dynamics

Coursework

- Course work (not graded)
- Grading:

  Project (90%)
  Wheel of misery (10%)

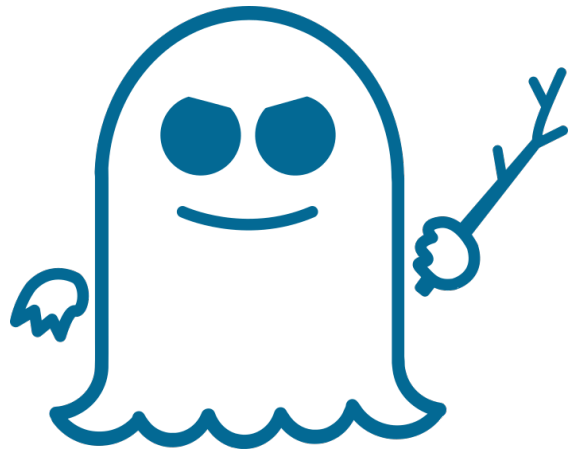
Day 1: take a laptop with GNU/Linux!

Importance of finding bugs

'worst ever' CPU bugs affect virtually all computers

S.Gibbs, The Guardian

Meltdown



Spectre

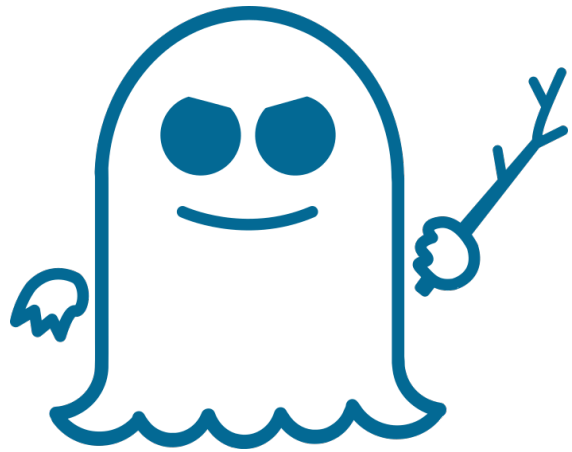
- Am I affected by the vulnerability? Most certainly, yes.
- Has Meltdown or Spectre been abused in the wild? We don't know.

Importance of finding bugs

'worst ever' CPU bugs affect virtually all computers

S.Gibbs, The Guardian

Meltdown



Spectre

> *"Industries and companies don't spend* 100 *billion dollars or devote these personnel resources to a problem they think is not serious"* – *20 Years Later, the Y2K Bug Seems Like a Joke*

Aim

Build correct tools to find bugs in code

```
1.  public class AClass {
2.  public static void main(String args[]) {
3.      int x = 1;
4.      int y = 3/(x++ - 1);
5.  }}
```

Aim

Build correct tools to find bugs in code

```
[ ...]$ analyse AClass.java
AClass.java:3:  error:  Divide By Zero
Expression '0' could be zero at line 3.

1.  public class AClass {
2.  public static void main(String args[]) {
3.  >    int x = 1;
4.       int y = 3/(x++ - 1);
5.  }}

Found 1 issue

  Issue Type(ISSUED_TYPE_ID): #
  Divide By Zero(DIVIDE_BY_ZERO): 1
```

Aim

Build sound languages to write bugless code

```
let _ =
"foo" + 42;;
```

Aim

Build sound languages to write bugless code

```
[ ...]$ ocaml aprog.ml
let _ =
"foo" + 42;;
  Line 2, characters 2-7:
  2 | "foo" > 42;;
      ^^^^^
 Error:  This expression has type string
 but an expression was expected of type int
```

Prerequisites 1

Know either OCaml or Haskell

Prerequisites 1

Know either OCaml or Haskell

Prerequisites 2

Know either OCaml or Haskell

Analyse Statique 2024-2025

Further details

- Git repository with teaching material: click here
- Course content outlined in files `syllabus.md` and `overview.pdf` in the repository
- Lectures either in English or French
- Research interships at SAP France, IMDEA Software, and at IRIF
- First lecture: January 20th 2024

Arrive *already* organised in team of *two* students.

Analyse Statique 2024-2025

Foundations order theory, results on fixed-points one lecture

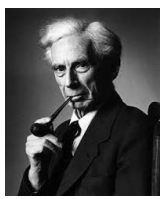
Dataflow Analysis

- WHILE$_{NNH}$
- Heart of optimizing compilers
- Objective: implement an analysis

Abstract Interpretation

- WHILE$_{RY}$
- INFER, ABSINT, MOPSA, FRAMA-C, ... written in OCAML
- Objective: implement an abstract interpreter

Along with historical remarks



or digressions on industrial tools.

Analyse Statique 2024-2025

https://fbinfer.com/

Final grade

100 % project

- implementation of monotone frameworks
  - generic framework
  - instance to perform constant propagation
- implementation of abstract interpretation

Introduction à l'apprentissage profond

Présentation du cours

Sam van Gool

IRIF, UFR d'informatique, Université Paris Cité

M2 informatique, 2024-2025, deuxième période

Qu'est-ce que l'apprentissage profond ?

- *Apprentissage automatique* : une approche statistique pour construire des programmes, également appelés *modèles*.
- *Réseaux de neurones profonds* : une catégorie particulière de modèles composés de plusieurs couches.
- Utilisés en « *intelligence artificielle* », particulièrement là où il est difficile de définir des règles déterministes :
  - Reconnaissance d'images ou de sons ;
  - Traduction de texte ;
  - Génération de code, de texte, d'images, etc.

Contenu du cours

Objectif

Comprendre les modèles d'apprentissage profond :

fonctionnement, *entraînement*, et *utilisation*.

- Fondements de l'apprentissage automatique
- Réseaux de neurones profonds
- Rétropropagation des gradients (*backprop*)
- Calcul automatique des gradients (*autograd*)
- Optimisation et régularisation pendant l'entraînement
- Réseaux de neurones convolutifs et récurrents
- Modèle transformer générateur (*GPT*)

Informations pratiques

- Ce cours combine :
  - *programmation* : Python orienté objet avec `numpy` et `pytorch` ;
  - *mathématiques* : dérivées, gradients, graphes, matrices, probabilités.
- 11 séances :
  - 1,5 h de cours magistral (S. van Gool)
  - 1,5 h de travaux pratiques (G. Baudart et S. van Gool)
- Modalités d'évaluation : TP notés (40 %) et examen (60 %).
- Livres de référence, disponibles en ligne gratuitement :
  - Prince, S., **Understanding Deep Learning**, `https://udlbook.com`
  - Zhang, A. et al., **Dive into Deep Learning**, `https://www.d2l.ai`

Transformation de programmes

Giulio Manzonetto

Transformation de programmes, késako ?

- Ancien nom : compilation avancée
- Accent sur les étapes *intermédiaires* d'un compilateur
- Ni parsing ni assembleur x86 (ou en option seulement), mais:
- Remaniement d'AST (arbres de syntaxe)
- Passages progressifs de langages haut-niveau à bas-niveau
- Comment rendre les récursivités terminales ?
- Qu'est-ce qu'une mise en CPS (Continuation Passing Style) ?

Un cours pour qui?

- Les étudiants qui ont bien aimé le cours de compilation
- Les étudiants qui ont trouvé le cours de compilation trop ‘bas niveau’ (on ne fera ni parsing ni assembleur x86)

Capacité d’accueil : 27

Prérequis

Plutôt des recommandations :

- Avoir suivi M1 Compilation est un plus
  - thème commun, même si on verra des techniques différentes
- Connaître au moins un langage fonctionnel
  - On programmera dans le fragment fonctionnel de Scala
- Avoir un ordinateur portable sur lequel travailler

Projet

Transformation d'un langage idéalisé mais représentatif (traits impératifs + fonctionnels) vers du bytecode JVM en assurant une récursivité efficace.

Evaluation

65% projet + 35% examen

Programmation Logique et Par Contraintes Avancée

Ralf Treinen

November 28, 2024

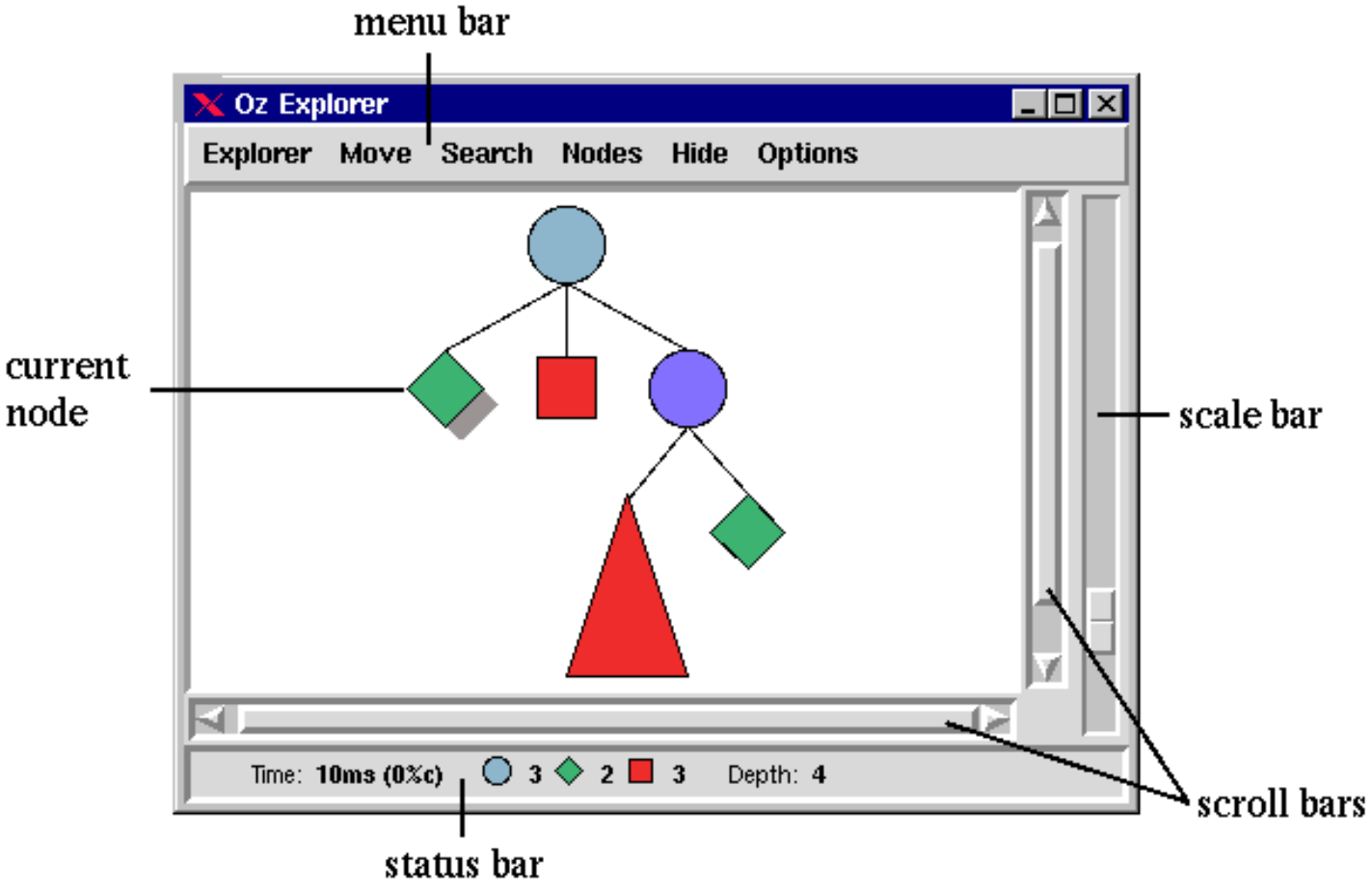
Programmation par contraintes : pourquoi ?

- Résoudre des problèmes qui demandent la *recherche* d'une solution
- Résolution de problèmes combinatoires:
  - Problèmes de planification
  - Problèmes d'ordonnancement optimal
  - Problèmes de placement optimal sous contraintes
  - Résolution des dépendances entre paquet Linux
  - Des puzzles ...

Programmation par contraintes, est-ce du Prolog ?

- Non !
- Nous allons programmer en *Oz* qui est un vrai langage de programmation.
- Oz permet de faire la programmation fonctionnelle, concurrente, (et impérative, orienté objet, des interfaces graphiques, de la distribution, ...)
- Construction d'un arbre de recherche peut être seulement une composante d'un programme complexe : la recherche est *encapsulée*.
- Mécanisme de calcul original, pas de Résolution.

Outil graphique pour interagir avec un arbre de recherche



Pre-requis

- Vous auriez besoin de :
  - Logique : logique propositionnelle et les bases de la logique du premier ordre, notion de base de la résolution de contraintes (simplification d'un système d'équations, par exemple)
  - Bases de la programmation fonctionnelle (OCaml, Haskell, Scala, F#, ...)
- Mais vous n'auriez **pas** besoin de :
  - Des connaissances de Prolog
  - Le cours de Programmation Logique du M1

Organisation du cours

- Format : 1.5h de cours, puis 1.5h de TP.
- Examen final (2h)
- Contrôle continu : TP noté dans la deuxième moitié du semestre
- Note : 50% CC + 50% Examen
- `http://www.irif.fr/~treinen/teaching/plpc/`

Algorithmique répartie

Chargée de cours: Lélia Blin (lelia.blin@irif.fr)

L'algorithmique répartie, c'est l'art de faire fonctionner en ensemble des entités de calculs autonomes. Dans ce cours nous découvrirons comment, les entités de calculs peuvent s'organisées entres elles pour résoudre des tâches tels que pratiqués dans les réseaux.

> Il est conseillé de suivre aussi le cours de M2 programmation distribuée.

Plan (sous-réserve)

- Introduction
- Horloges
- Communications
- Etats globaux consistants
- Election
- Partage de ressources
- Construction d'arbres couvrants
- Tolérance aux fautes transitoires

Evaluation

- 2 devoirs
- Un examen final

Note finale: moyenne des 3

Grands Réseaux d'Interaction

Vlady Ravelomanana

IRIF – Univ. Paris
vlad@irif.fr

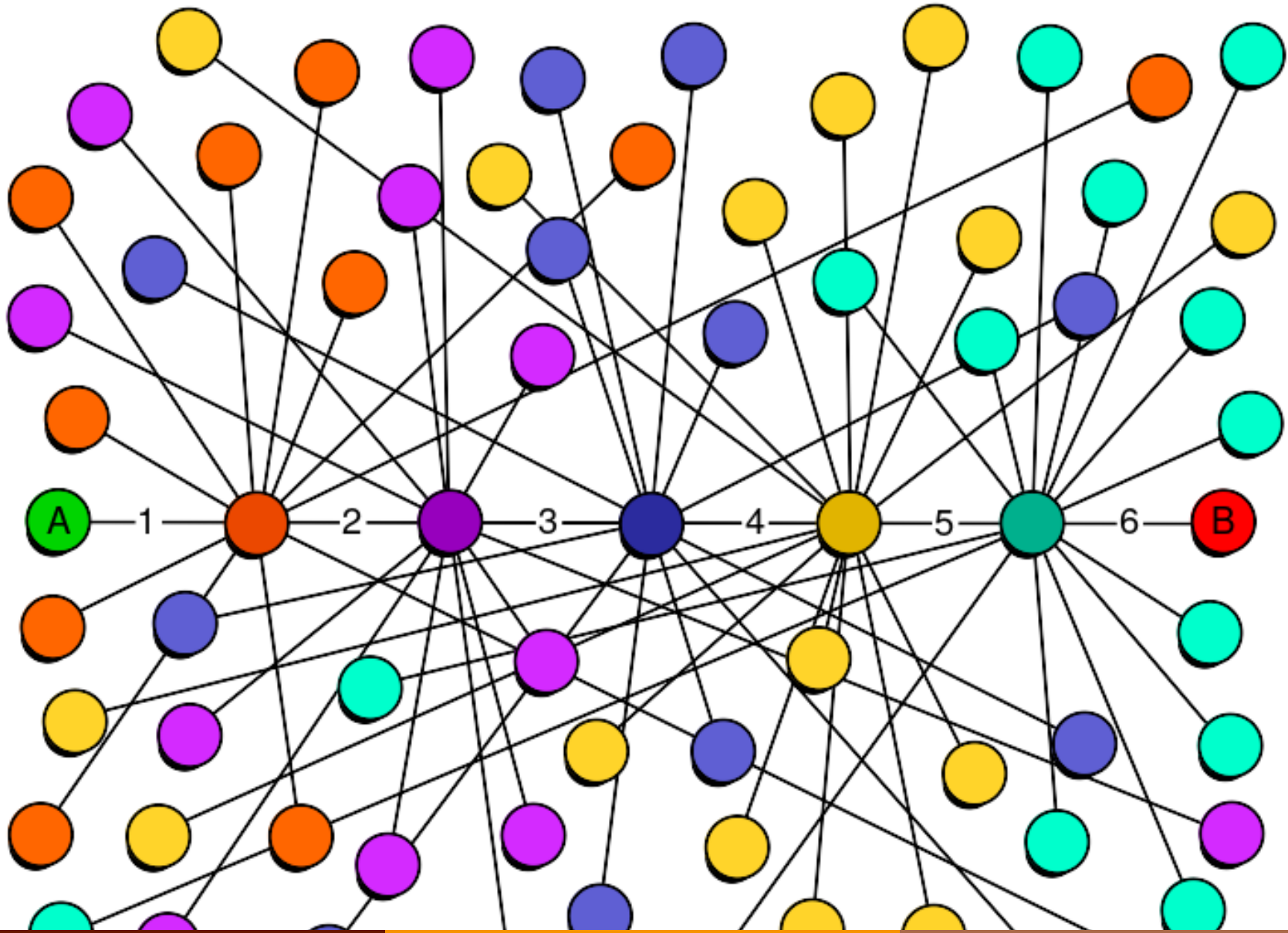
Université de Paris Cité

— Transparents : Présentation du Cours —

Plan du cours

1. DESCRIPTION
   - Comprendre les propriétés structurelles des GRI en informatique, sociologie, biologie, physique, linguistique...
   - $\longrightarrow$ **En TP :** Calcul de divers paramètres
   - Big data – structures de données et algorithmes efficaces
2. MODÉLISER
   - Classifier ces réseaux d'interaction
   - Modèles aléatoires $\longrightarrow$ Si mêmes propriétés que le réel, alors même génération ?
   - $\longrightarrow$ **En TP :** Génération aléatoire
3. APPLICATIONS
   - Dissémination (contagion ou broadcast/gossip)
   - P2P (diffusion temps réel)

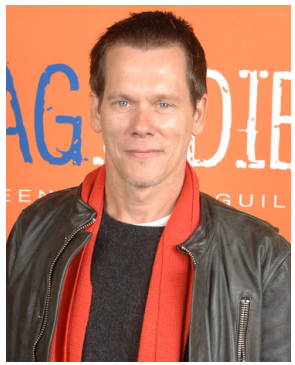
Une vieille idée / théorie



Ideés

- Hypothèse proposée et popularisée par l'écrivain hongrois Frigyes Karinthy [a] en 1929.
- Toute chaîne de relations individuelles fait au plus six maillons.
- Cette hypothèse était dans l'air du temps au début du siècle dernier.

a. «En humour, je ne plaisante jamais.»



Six Degrees of Kevin Bacon (« Les Six degrés de Kevin Bacon » en français) est un jeu de quiz s'appuyant sur le concept du phénomène du petit monde et l'hypothèse selon laquelle chaque acteur peut être relié à Kevin Bacon via ses rôles dans six films ou moins. Le nom du jeu vient de la pièce de théâtre ayant illustré le principe des six degrés de séparation (*six degrees of separation*).

En 2007, Bacon a créé une organisation de charité nommée *SixDegrees.org*.

Stanley Milgram 1933–1984, psychologue social américain

Très connu pour ses expériences de soumission à l'autorité

En 1967, Milgram a essayé de valider expérimentalement les six degrés de séparation : l'étude du **petit monde**.

L'expérience de Milgram (1963)

- origine : 60 personnes dans la ville d'Omaha, Nebraska.
- cible : agent de change vivant à Sharon, Massachusetts.
- règle : envoyer une lettre à quelqu'un que l'on appelle par le prénom ($\simeq$ à un ami) qui va la **relayer**
- transmission manuelle (et non postale) des lettres
- Le célèbre article de 1967 de Milgram décrit le fait qu'une lettre ne mit que quatre jours pour atteindre sa destination
- mais négligea de mentionner que seulement 5 % des lettres (**3 lettres !**) réussirent à rejoindre leur cible.
- L'expérience a été refaite peu après avec 296 lettres. Seules 64 lettres sont arrivées, avec effectivement ≤ 6 sauts *en moyenne*

Le monde est petit ...

- La distance moyenne de Facebook est de 4.74
- À cause de l'invariance d'échelle il est crédible que celle du «vrai monde social» ne soit pas beaucoup plus, $\leqslant 6$.
- Entre les puissants de ce monde, mieux connectés, on peut espérer une plus petite distance encore ...

1. **Facebook** (188 millions de likes)
2. **Cristiano Ronaldo** (168 millions de likes)
3. **Samsung** (161 millions de likes)
4. **Mr Bean** (140 millions de likes)
5. **5-Minute Crafts** (126 millions de likes)
6. **Shakira** (124 millions de likes)
7. **Real Madrid C.F.** (121 millions de likes)
8. **CGTN** (121 millions de likes)
9. **Will Smith** (116 millions de likes)
10. **Lionel Messi** (116 millions de likes)

Principaux Objectifs Algorithmiques dans les Réseaux d'Interaction

1. **Détection et Analyse des Communautés**
   - Identifier des groupes fortement interconnectés.
   - Techniques : Algorithme Louvain, modularité.
2. **Analyse de la Centralité**
   - Identifier les nœuds influents dans un réseau.
   - Mesures : Degré, centralité d'intermédiarité, autovecteurs.
3. **Optimisation des Flux**
   - Maximiser la diffusion ou minimiser les coûts.
   - Algorithmes : Flot maximal, Dijkstra.
4. **Prédiction de Lien**
   - Anticiper de nouvelles connexions.
   - Techniques : Indice de Jaccard, apprentissage machine.
5. **Propagation et Diffusion**
   - Étudier la propagation d'idées, virus ou rumeurs.
   - Modèles : SIR, modèles de seuil.
6. **Robustesse et Résilience**
   - Évaluer la résistance aux pannes ou attaques.
   - Analyse des cascades de défaillance.
7. **Recommandation Personnalisée**
   - Proposer du contenu ou des connexions pertinentes.
   - Techniques : Filtrage, parcours aléatoire.
8. **Identification des Anomalies**
   - Détecter des comportements inhabituels.
   - Méthodes : Clustering, ...
9. **etc ... etc ... etc ...**

Le Modèle Jouet : Erdos-Rényi $G(n, p)$

- **Définition de $G(n, p)$:**
  Un **graphe aléatoire** où chaque paire de nœuds est connectée par une arête avec probabilité p.
  Par exemple, pour un graphe avec $n = 5$ et $p = 0.3$, chaque arête a 30% de chance d'exister.
- **Propriétés du Modèle :**
  - Les degrés des nœuds suivent une *loi binomiale*.
  - Propriétés attendues :
    (1) La connectivité *moyenne* dépend de n et de p.
    (2) Petits composants : des sous-graphes indépendants apparaissent lorsque p est faible.
- **Limites de $G(n, p)$:**
  - Ne capture pas les structures complexes comme *small-world* ou *scale-free*.
  - Pas de *corrélations* dans les degrés des nœuds (les arêtes sont indépendantes).

Principales Difficultés dans les Grands Réseaux

1. **Taille et Complexité**
   - Les réseaux réels peuvent contenir des millions de nœuds et connexions.
   - Problème : Difficulté à stocker et traiter ces données à grande échelle.
2. **Modélisation Réaliste**
   - Capturer les propriétés des réseaux réels (e.g., distribution des degrés, modularité).
   - Limitation : Le modèle $G(n, p)$ génère des graphes aléatoires avec une distribution uniforme des degrés.
3. **Structures Complexes**
   - Les réseaux réels contiennent des motifs spécifiques : communautés, hubs.
   - $G(n, p)$ ne capture ni les communautés ni les corrélations locales.
4. **Pourquoi $G(n, p)$ échoue**
   - Absence de **hétérogénéité** dans les degrés des nœuds.
   - Aucun mécanisme pour générer des propriétés comme **small-world** ou **scale-free**.
5. **Modèles Alternatifs**
   - **Modèle Small-World** : (Watts-Strogatz) pour capturer la connectivité locale + distances courtes.
   - **Modèle Scale-Free** : (Barabási-Albert) pour simuler des réseaux avec distribution de degré en loi de puissance.
   - **Graphes Stochastiques Bloqués** : Pour les réseaux avec communautés distinctes.
6. **Comment aborder ces défis ?**
   - **Données massives** : Algorithmes distribués et stockage efficace.
   - **Validation empirique** : Comparer les propriétés du modèle avec celles des réseaux réels.
   - **Hybridation** : Combiner plusieurs modèles pour répondre aux contraintes spécifiques.

Évaluation/MCC

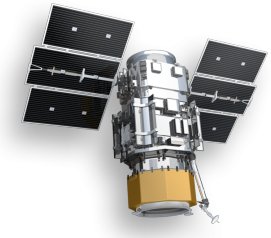
- **Un examen pour 1/3**
- **Aborder les défis des GRI par la pratique pour 2/3**

  TP obligatoires (python/java/ ...) et 2 mini-projets individuels :

Mobilité

Matěj Stehlík

matej@irif.fr



Aperçu du cours

- Nous étudierons des problèmes et algorithmes liés à la mobilité, comme par exemple :
  - le calcul d'itinéraire
  - l'optimisation du trafic dans un réseau
  - l'allocation de fréquences
  - les algorithmes *online*
- Il s'agit d'un cours *théorique*, qui s'appuie beaucoup sur la notion des *graphes* et parfois aussi sur la théorie des jeux.
- Le but est de donner des bases théoriques des différents algorithmes, mais il y aura aussi du travail sur l'ordinateur pour s'approprier ces concepts, et pour implémenter les algorithmes.

Vous apprendrez. . .

- que trouver le meilleur itinéraire est (beaucoup) plus complexe pour les livreurs que pour les postiers.
- de créer un planning de vols pour une flotte d'avions.
- d'attribuer des fréquences aux antennes d'un réseau cellulaire.
- que (paradoxalement) fermer une route peut améliorer le trafic dans une ville !

Informations complémentaires

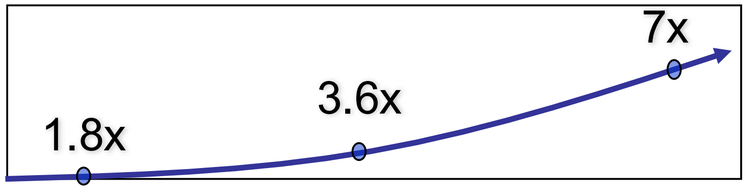
- Pré-requis théoriques : graphes (on fera un rappel des notions de base lors du premier cours)
- Pré-requis pratiques : Python ou Java
- Apportez vos propres ordinateurs.
- L'évaluation sera basée sur une épreuve écrite et éventuellement un projet/devoir maison.

Programmation répartie

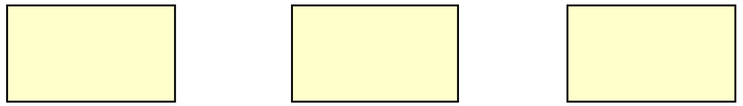
Vendredi 8h30-11h30

Carole Delporte cd@irif.fr

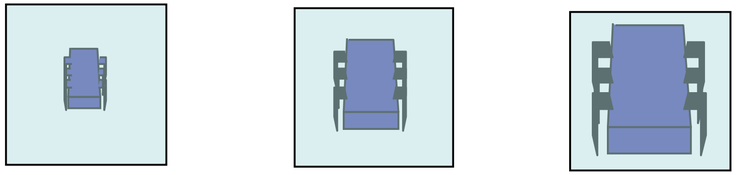
Accélaration



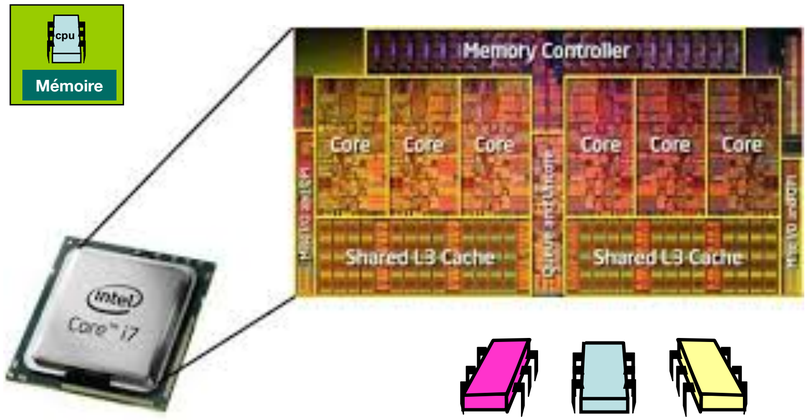
Code utilisateur

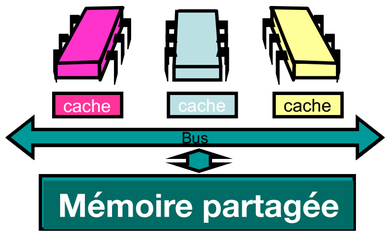


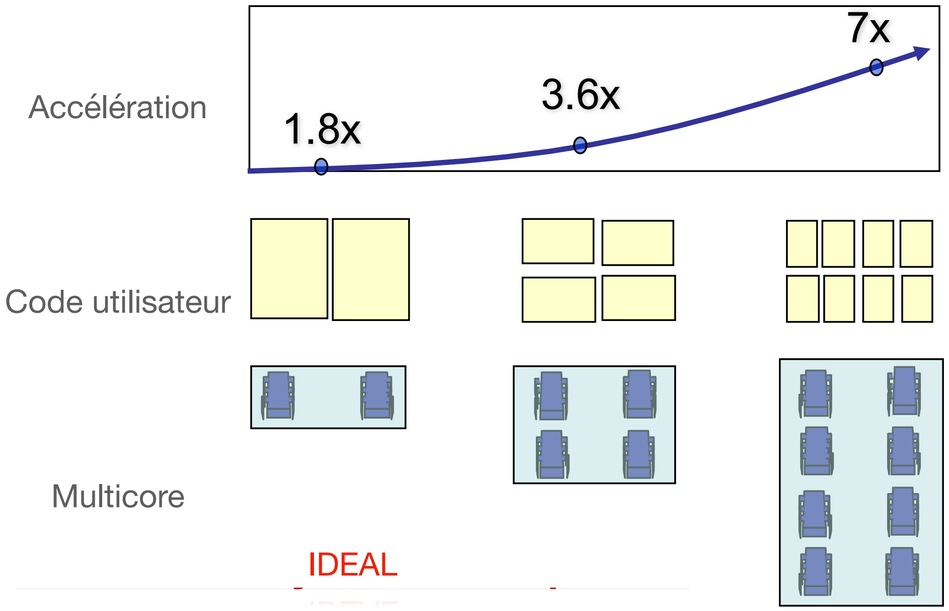
Multicore

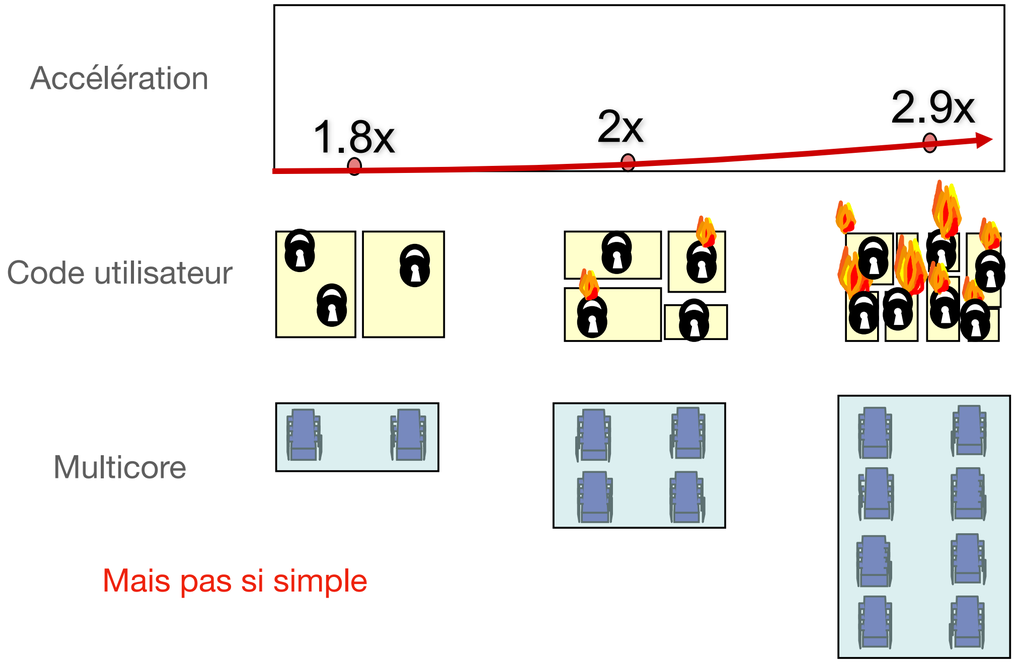


Temps : Loi de Moore









Apprenez les principes fondamentaux de la programmation de plusieurs threads accédant à la mémoire partagée, des simples verrous aux primitives plus évoluées

Programmez (en Java) des structures de données concurrentes

Organisation et Controle des connaissances

- Cours et TP
- TP regroupés par thème notés (sur 10)
- Examen écrit la semaine des examens (sur 10)
- Note finale =note exam +note TP

Le Stage

- Durée minimale de 4 mois.
- Durée maximale de 6 mois (contrainte légale)
- Entre le 31 mars et le 30 septembre.
- Rapport de stage, soutenance (normalement publique) en septembre.
- Stage en entreprise (aussi startup) ou Stage de recherche.
- Stage à l'étranger possible, mais procédure administrative particulière.

Trouver son Stage

- **Le sujet doit être approuvé par le responsable du parcours.**
- Rédaction et signature de la convention : voir `http://www.informatique.univ-paris-diderot.fr/formations/masters/m2_stages_fin_etudes`
- La signature de la convention prend du temps (en particulier pour des organismes sensibles à la sécurité).
- Commencez tôt à vous en occuper.
- Nous transmettons des offres de stages sur la liste de diffusion des parcours concernés ; vérifiez votre adresse email renseignée sur silice.

Informations et contacts

- Le wiki de l'UFR : `http://www.informatique.univ-paris-diderot.fr`
- Serveur pédagogique : `https://silice.informatique.univ-paris-diderot.fr/ufr`
- Suivi administratif : Sylvia CROCHET
  bureau Sophie-Germain 3002
  `crochet@informatique.univ-paris-diderot.fr`
- Responsables de parcours : prendre rendez-vous par email
  - DATA : Amélie Gheerbrant `Amelie.Gheerbrant@irif.fr`
  - IMPAIRS : Carole Delporte `Carole.Delporte@irif.fr`
  - LP : Giulio Manzonetto `gmanzone@irif.fr`
    et Ralf Treinen `treinen@irif.fr`
- Ingénieur : Laurent Pietroni
  `pietroni@informatique.univ-paris-diderot.fr`