Séminaire interne MIAT
15 octobre 2021

Résoudre un problème d'optimisation exprimé par un réseau de fonctions de coûts avec toulbar2 et les services Web

Simon de Givry
équipe SaAB

Nathalie Rousse
équipe RECORD

A. Problème d’optimisation combinatoire

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

A. Problème d'optimisation combinatoire

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

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

A. Problème d’optimisation combinatoire

X1 9	X2 -	X3 8	X4 -	X5 7	X6 5	X7 -	X8 1	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	...	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)

A. Problème d'optimisation combinatoire

X1 9	X2 -	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	...	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine

A. Problème d’optimisation combinatoire

X1 9	X2 -	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	... -	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 27 contraintes de permutation

AllDifferent(X1,X2,X3,X4,X5,X6,X7,X8,X9)

AllDifferent(X10,X11,X12,X13,X14,X15,X16,X17,X18)

…

AllDifferent(X1,X2,X3,X10,X11,X12,X19,X20,X21)

AllDifferent(X4,X5,X6,X13,X14,X15,X22,X23,X24)

…

AllDifferent(X1,X10,X19,X28,X37,X46,X55,X64,X73)

AllDifferent(X2,X11,X20,X29,X38,X47,X56,X65,X74)

...

A. Problème d'optimisation combinatoire

X1 9	X2 1	X3 8	X4 -	X5 7	X6 5	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	... -	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 27 contraintes de permutation

Arité 9

AllDifferent(X1,X2,X3,X4,X5,X6,X7,X8,X9)

AllDifferent(X10,X11,X12,X13,X14,X15,X16,X17,X18)

...

AllDifferent(X1,X2,X3,X10,X11,X12,X19,X20,X21)

AllDifferent(X4,X5,X6,X13,X14,X15,X22,X23,X24)

...

AllDifferent(X1,X10,X19,X28,X37,X46,X55,X64,X73)

AllDifferent(X2,X11,X20,X29,X38,X47,X56,X65,X74)

...

A. Problème d’optimisation combinatoire

X1 9	X2 1	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	... -	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6 ... X81

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence

X1≠X2, X1≠X3, X1≠X4, X1≠X5, X1≠X6, X1≠X7, X1≠X8, X1≠X9, X2≠X3, X2≠X4,…

…

X1≠X10, X1≠X11, X1≠X12, X1≠X19, X1≠X20, X1≠X21, X2≠X10, X2≠X11,...

...

X1≠X19, X1≠X28…

...

A. Problème d'optimisation combinatoire

X1 9	X2 1	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	... -	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6 ... X81

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence

Arité 2

X1≠X2, X1≠X3, X1≠X4, X1≠X5, X1≠X6, X1≠X7, X1≠X8, X1≠X9, X2≠X3, X2≠X4,…

…

X1≠X10, X1≠X11, X1≠X12, X1≠X19, X1≠X20, X1≠X21, X2≠X10, X2≠X11,...

...

X1≠X19, X1≠X28…

...

A. Problème d’optimisation combinatoire

X1 9	X2 -	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	...	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 30 contraintes d’égalité

X1=9, X3=8, X5=7,X8=4, X9=2, X13=9,...

A. Problème d’optimisation combinatoire

X1 9	X2 -	X3 8	X4 -	X5 7	X6 -	X7 -	X8 4	X9 2
X10 -	X11 -	X12 -	X13 9	X14 -	X15 -	X16 -	X17 6	X18 -
X19 -	X20 -	X21 -	... -	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6

... X81

Formulation

Il s’agit d’exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 30 contraintes d’égalité
- *pas de fonction objectif (problème de satisfaction de contraintes)*

A. Problème d'optimisation combinatoire

X1 9	X2 1	X3 8	X4 5	X5 7	X6 6	X7 3	X8 4	X9 2
X10 4	X11 3	X12 5	X13 9	X14 2	X15 1	X16 7	X17 6	X18 8
X19 6	X20 7	X21 2	...	-	-	-	-	-
-	-	6	8	-	4	-	9	-
5	8	3	-	-	-	-	-	4
-	-	-	-	6	3	-	-	7
3	5	-	-	-	2	8	-	-
-	6	-	-	3	7	-	-	9
2	-	7	-	-	5	-	-	6 ... X81

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes

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

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 30 contraintes d'égalité
- *pas de fonction objectif (problème de satisfaction de contraintes)*

Solution : affectation complète qui satisfait toutes les contraintes

X1=9, X2=1, X3=8, X4=5, X5=7, X6=6,…, X81=6

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 81 fonctions de score (CNN LeNet)

f(X1)=(0.11, 0.11,…, 0.11)

...

f(X5)=(0, 0, 0, 0, 0, 0, 0, 1, 0)

...

f(X7)=(0.25, 0, 0, 0.05, 0, 0, 0.7, 0, 0)

...

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 81 fonctions de coûts

f(X1)=(0, 0, 0, 0, 0, 0, 0, 0, 0)

...

f(X5)=(T, T, T, T, T, T, T, 0, T)

...

f(X7)=(139, T, T, 300, T, T, 36, T, T)

...

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 81 fonctions de coûts

f(X1)=(0, 0, 0, 0, 0, 0, 0, 0, 0)

...

f(X5)=(T, T, T, T, T, T, T, 0, T)

T : coût infini (affectation interdite)

...

f(X7)=(139, T, T, 300, T, T, 36, T, T)

...

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes
- fonction objectif

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 contraintes de différence
- 81 fonctions de coûts

f(X1)=(0, 0, 0, 0, 0, 0, 0, 0, 0)

...

f(X5)=(T, T, T, T, T, T, T, 0, T)

...

f(X7)=(139, T, T, 300, T, T, 36, T, T)

-log(0.25)

...

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- fonctions de coûts

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 fonctions de coûts binaires

f(X1,X2)=(T, 0, 0, 0, 0, 0, 0, 0, 0)
(0, T, 0, 0, 0, 0, 0, 0, 0)
(0, 0, T, 0, 0, 0, 0, 0, 0)
(0, 0, 0, T, 0, 0, 0, 0, 0)
(0, 0, 0, 0, T, 0, 0, 0, 0)
(0, 0, 0, 0, 0, T, 0, 0, 0)
(0, 0, 0, 0, 0, 0, T, 0, 0)
(0, 0, 0, 0, 0, 0, 0, T, 0)
(0, 0, 0, 0, 0, 0, 0, 0, T)

- 81 fonctions de coûts unaires

Réseau de fonctions de coûts

Trouver une affectation complète qui minimise la somme des fonctions de coûts (NP-dur)

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- fonctions de coûts

- 81 variables (cellules)
- 9 valeurs par domaine
- 810 fonctions de coûts binaires

f(X1,X2)=(T, 0, 0, 0, 0, 0, 0, 0, 0)
(0, T, 0, 0, 0, 0, 0, 0, 0)
(0, 0, T, 0, 0, 0, 0, 0, 0)
(0, 0, 0, T, 0, 0, 0, 0, 0)
(0, 0, 0, 0, T, 0, 0, 0, 0)
(0, 0, 0, 0, 0, T, 0, 0, 0)
(0, 0, 0, 0, 0, 0, T, 0, 0)
(0, 0, 0, 0, 0, 0, 0, T, 0)
(0, 0, 0, 0, 0, 0, 0, 0, T)

- 81 fonctions de coûts unaires

Réseau de fonctions de coûts

Trouver une affectation complète qui minimise la somme des fonctions de coûts (NP-dur)

→ solveur **toulbar2**
https://miat.inrae.fr/toulbar2

A. Problème d'optimisation combinatoire

X1 C1	X2 C2	X3 C3	X4 C4	X5 C5	X6 C6	X7 C7	X8 C8	X9 C9
4	9	1		6			2	8
5			3	4		1		
		3		7	9		1	
1	7					5		
	5					9	6	
	6	2	1		7		8	
	3				8	2	5	
8					4			

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables de décision
- domaines finis de valeurs
- contraintes

Programmation par contraintes

Trouver une affectation complète qui minimise la variable objectif C

→ solveur **MiniZinc**
https://www.minizinc.org

- 163 variables (Xi, Ci et C)
- 9 valeurs par domaine pour chaque Xi, un entier positif [0,T] pour Ci et C
- 27 contraintes *AllDifferent*
- 81 contraintes *Element*

…

C7=Element(X7, (139, T, T, 300, T, T, 36, T, T))

...

- 1 contrainte pour la variable objectif

C = C1 + C2 + C3 + … + C81

A. Problème d'optimisation combinatoire

Formulation

Il s'agit d'exprimer un problème à résoudre sous forme de :

- variables 0/1
- contraintes linéaires
- fonction objectif linéaire

Programmation linéaire en nombres 0/1

Trouver une affectation complète qui minimise la fonction objectif

- 729 variables (Xi,j : cellule i, chiffre j)
- 2 valeurs par domaine {0,1}
- 81 contraintes de domaine entier

Pour tout i, Xi,1 + Xi,2 + Xi,3 + … + Xi,9 = 1

- 243 contraintes de permutation

Pour tout j, X1,j + X2,j + X3,j + … + X9,j = 1

- 1 fonction objectif linéaire

*… + 139*X7,1 + T*X7,2 + T*X7,3 + 300*X7,4 + T*X7,5+ T*X7,6 + 36*X7,7 +...*

→ solveur **cplex**
https://www.ibm.com/analytics/cplex-optimizer

B. Exemple du problème de conception d'un emploi du temps dans une université

Le problème :

Il s’agit de programmer (dates et salles)

une session d’examens en fin de semestre

sur un campus universitaire.

La publication : **17th International Conference, CPAIOR 2020**

(Constraint Programming, Artificial Intelligence, and Operations Research)

Vienna, Austria, September 21–24, 2020

Local Search and Constraint Programming for a Real-World Examination Timetabling Problem

Michele Battistutta (1) , Sara Ceschia (2) , Fabio De Cesco (1) , Luca Di Gaspero (2) ,
Andrea Schaerf (2) , and Elena Topan (1)
(1) EasyStaff srl, Via Adriatica, 278, 33030 Campoformido, UD, Italy
(2) DPIA, University of Udine, Via delle scienze 206, 33100 Udine, Italy

Formulation du problème au travers de contraintes et d'objectifs.

Collection d'instances du problème

Approche métaheuristique

→ Recherche Locale avec Recuit Simulé

Programmation par Contraintes (CSP)

→ Modèle MiniZinc

Résultats

Comparaison

La publication : **17th International Conference, CPAIOR 2020**

(Constraint Programming, Artificial Intelligence, and Operations Research)

Vienna, Austria, September 21–24, 2020

Local Search and Constraint Programming for a Real-World Examination Timetabling Problem

Michele Battistutta (1) , Sara Ceschia (2) , Fabio De Cesco (1) , Luca Di Gaspero (2) ,
Andrea Schaerf (2) , and Elena Topan (1)
(1) EasyStaff srl, Via Adriatica, 278, 33030 Campoformido, UD, Italy
(2) DPIA, University of Udine, Via delle scienze 206, 33100 Udine, Italy

Formulation du problème au travers de contraintes et d'objectifs.

Collection d'**instances du problème**

Approche métaheuristique

→ Recherche Locale avec Recuit Simulé

Programmation par Contraintes (CSP)

→ **Modèle MiniZinc**

Réseau de fonctions de coûts (WCSP)

→ **Modèle toulbar2**

Résultats

Comparaison

Formulation du problème :

Event

1 **Course** ↔ 1 or more **Exam**s

1 **Exam** ↔ *'W'* **Event**
or *'O'* **Event**
or *'W'* **Event** then *'O'* **Event**

Room

1 **Event** ↔ 0 or 1 or more **Room**s

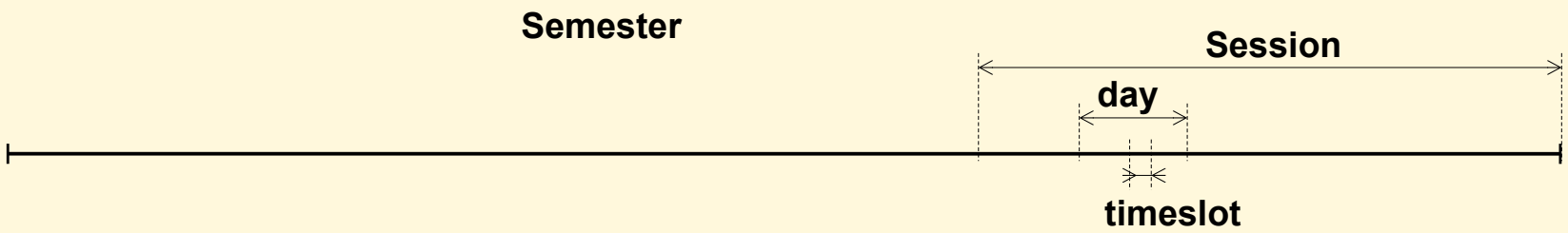
(0 : teacher's office, external rooms)

Room <u>size</u> : 'Small', 'Medium' or 'Large'

1 **Roomset** ↔ 1 or more Rooms of same <u>size</u>

1 *'W'* **Event** ↔ a **Roomset**
1 *'O'* **Event** ↔ 0 or 1 **Room**

Period



=> Le problème : Il s'agit d'attribuer à chaque **Event** : une **Period** et une **Room** (*1 Room, 1 Roomset, ou aucune*)

tout en satisfaisant un certain nombre de **contraintes** et de **préférences**.

Illustration « toy instance »

COURSES

Name	Teacher	#Exams	Exam Distance	Exam Type	$\mathcal{W}/\mathcal{O}$ Distance	Location
Databases	Andrea	2	5	$\mathcal{W}+\mathcal{O}$	(4,8)	2 Large
Algorithms	Luca	2	5	$\mathcal{W}$	—	1 Large
Program.	Elena	2	4	$\mathcal{W}$	—	2 Large
Oper. Res.	Sara	2	4	$\mathcal{W}+\mathcal{O}$	(1,1)	1 Small + 1 Small
Analytics	Michele	1	—	$\mathcal{O}$	—	—

ROOMS

Name	Type
A	Large
B	Large
C	Large
G	Small
H	Small

ROOMSETS

Roomset	Size	Type
{A, B}	2	Large
{A, C}	2	Large

PERIODS

20

*Une **Period** peut valoir :* *1, … , 20*

*Une **Room** peut valoir :* *'A', 'B', 'C', 'G', 'H', 'A,B', 'A,C', '–'*

Illustration « toy instance »

COURSES

Name	Teacher	#Exams	Exam Distance	Exam Type	$\mathcal{W}/\mathcal{O}$ Distance	Location
Databases	Andrea	2	5	$\mathcal{W}+\mathcal{O}$	(4,8)	2 Large
Algorithms	Luca	2	5	$\mathcal{W}$	—	1 Large
Program.	Elena	2	4	$\mathcal{W}$	—	2 Large
Oper. Res.	Sara	2	4	$\mathcal{W}+\mathcal{O}$	(1,1)	1 Small + 1 Small
Analytics	Michele	1	—	$\mathcal{O}$	—	—

ROOMS

Name	Type
A	Large
B	Large
C	Large
G	Small
H	Small

ROOMSETS

Roomset	Size	Type
{A, B}	2	Large
{A, C}	2	Large

PERIODS

20

SOLUTION

Event				Assignment	
Course	Exam	Part	ID	Period	Rooms
Databases	#1	$\mathcal{W}$	0	2	A, B
		$\mathcal{O}$	1	5	—
	#2	$\mathcal{W}$	2	12	A, B
		$\mathcal{O}$	3	15	—
Algorithms	#1	$\mathcal{W}$	4	0	B
	#2	$\mathcal{W}$	5	10	A
Programming	#1	$\mathcal{W}$	6	0	A, C
	#2	$\mathcal{W}$	7	17	A, B
Oper. Res.	#1	$\mathcal{W}$	8	6	H
		$\mathcal{O}$	9	7	G
	#2	$\mathcal{W}$	10	16	H
		$\mathcal{O}$	11	17	G
Analytics	#1	$\mathcal{O}$	12	8	—

Pour i = 0,…,12 :

variables de décision ***domaines de valeurs***

Period[i] ∈ *{ 1, … , 20 }*

Room[i] ∈ *{ 'A', 'B', 'C', 'G', 'H', 'A,B', 'A,C', '–' }*

Formulation du problème :

Contraintes (obligations, interdictions) - Hard Constraints -

- **H1. RoomRequest** : type et nombre de Rooms à respecter pour chaque Event.
- **H2. RoomOccupation** : à chaque Period, un Event maximum par Room.
- **H3. HardConflicts** : certains Events interdits en même temps (à même Period) ; exemple : si même teacher.
- **H4. Precedences** : chronologie imposée entre certains Events ; exemple : *'W'* avant *'O'* pour un même Exam.
- **H5 : Unavailabilities** : certains Events ou Rooms interdits à certaines Periods.

Préférences (objectifs) - Soft Constraints -

- **S1. SoftConflicts** : préférence pour que certains Events ne soient pas en même temps (à même Period).
- **S2. Preferences** : préférence de certains Events à certaines Periods, de certaines Rooms à certaines Periods.
- **S3. Distances** : préférence pour une certaine distance (en terme de Periods) entre certains Events ; exemple : distances min et max entre *'W'* et *'O'* pour un même Exam.

Implémentation avec MiniZinc :

MiniZinc est un **langage de modélisation par contraintes**, permettant de modéliser des problèmes d'optimisation et de satisfaction de contraintes.

Modèle **model.mzn**

Expression des contraintes et préférences

H1, H2…
S1, S2…

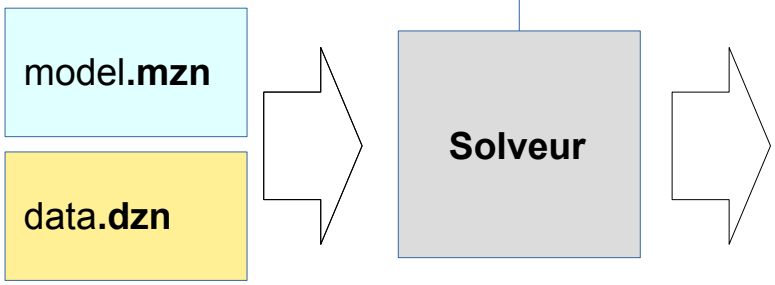
Résolution

Données **data.dzn**

Instanciation du cas d'utilisation

Nombre de Rooms et Periods,caractéristiques des Cours (besoins en Rooms…) ...

Plusieurs **solveurs** capables de résoudre un problème programmé sous **MiniZinc** : **chuffed**, **gecode**, **coin-bc**, **cplex**, **OR-Tools**...



EventPeriod et EventRoom de chaque Event

et Coût des préférences

MiniZinc : www.minizinc.org , https://slidetodoc.com/an-introduction-to-minizinc-minizinc-mini-zinc-is

Implémentation avec MiniZinc :

Event			
Course	Exam	Part	ID
Databases	#1	$\mathcal{W}$	0
		$\mathcal{O}$	1
	#2	$\mathcal{W}$	2
		$\mathcal{O}$	3
Algorithms	#1	$\mathcal{W}$	4
	#2	$\mathcal{W}$	5
Programming	#1	$\mathcal{W}$	6
	#2	$\mathcal{W}$	7
Oper. Res.	#1	$\mathcal{W}$	8
		$\mathcal{O}$	9
	#2	$\mathcal{W}$	10
		$\mathcal{O}$	11
Analytics	#1	$\mathcal{O}$	12

```
EventRoomConstraints =
 [ |-1, -1, -1, -1, -1,  0,  0, -1
   |-1, -1, -1, -1, -1, -1, -1,  0
   |-1, -1, -1, -1, -1,  0,  0, -1
   |-1, -1, -1, -1, -1, -1, -1,  0
   | 0,  0,  0, -1, -1, -1, -1, -1
   | 0,  0,  0, -1, -1, -1, -1, -1
   |-1, -1, -1, -1, -1,  0,  0, -1
   |-1, -1, -1, -1, -1,  0,  0, -1
   | 0,  0,  0,  0,  0, -1, -1, -1
   | 0,  0,  0,  0,  0, -1, -1, -1
   | 0,  0,  0,  0,  0, -1, -1, -1
   | 0,  0,  0,  0,  0, -1, -1, -1
   |-1, -1, -1, -1, -1, -1, -1,  0|];
```

```
EventPeriodConstraints =
 [|-1, 0, 0,  0, -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   |-1, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   |-1, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   |-1, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0, 10, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0, 10, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
   | 2, 2, 2,  0, -1, 2, 2, 2, 2, 10, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
   | 0, 0, 0,  0, -1, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0|];
```

A	Large
B	Large
C	Large
G	Small
H	Small

{A, B}	2	Large
{A, C}	2	Large

(no room) 'dummy'

PERIODS

20

- **H1**

COURSES

Name	Location
Databases	2 Large
Algorithms	1 Large
Program.	2 Large
Oper. Res.	1 Small + 1 Small
Analytics	—

- **H5**

-1 Forbidden < *0 desired allowed* < *2 prefered but not as desired* < *10 undesired*

- **S2**

Implémentation avec MiniZinc :

Données toy.dzn

Events = 13;
Periods = 20;
Rooms = 8;

EventRoomConstraints = …
EventPeriodConstraints = …
…

Modèle exam_tt.mzn

Variables Period et Room recherchées pour chaque Event

Event : [1, 2, . . . , Events] — *e ∈ { 1, … , Events }*
EventPeriod : [. . .] — *EventPeriod[e] ∈ { 1, … , Periods }*
EventRoom : [. . .] — *EventRoom[e] ∈ { 1, … , Rooms }*

Contraintes

```
forall(e in 1..Events, r in 1..Rooms where EventRoomConstraints[e, r] = -1)
    (EventRoom[e] != r);

forall(e in 1..Events, p in 1..Periods where EventPeriodConstraints[e, p] = -1)
    (EventPeriod[e] != p);
…
```

H1

H5

Coûts des préférences

```
PeriodPreferenceCost = sum(e in 1..Events)(EventPeriodConstraints[e, EventPeriod[e]]);
…
```

S2

```
solve minimize … + PeriodPreferenceCost + … ;
```

Résoudre en minimisant la somme des coûts

Implémentation avec toulbar2:

toulbar2 est un solveur pour réseau de fonctions de coût.

Problème codé en **C++** ou en **Python**

Instanciation du cas d'utilisation
Nombre de Rooms et Periods,caractéristiques des Cours (besoins en Rooms…) …
Expression des contraintes et préférences
H1, H2…
S1, S2…

Résolution

toulbar2 est programmé en langage **C++** et dispose d'une interface **Python pytoulbar2** (installation : *pip install pytoulbar2*)

En Python :

problem.py

*from **pytoulbar2** import …*

...

'python3 problem.py'

EventPeriod et EventRoom de chaque Event

et Coût total du réseau des fonctions de coût

toulbar2: https://miat.inrae.fr/toulbar2

Implémentation avec MiniZinc → Implémentation avec toulbar2

toy.dzn

Events = 13;
Periods = 20;
Rooms = 8;

EventRoomConstraints = …
EventPeriodConstraints = …
…

dzn to py

exam_tt.py

Events = 13
Periods = 20
Rooms = 8

EventRoomConstraints = […]
EventPeriodConstraints = […]
…

exam_tt.mzn

EventPeriod **EventRoom**

Contraintes
H1
…
H5

Préférences
…
S2

solve minimize … + … ;

EventPeriod **EventRoom**

Fonctions de coûts

Solve()

Event	: [1, 2, . . . , Events]	$e \in \{1, \ldots, Events\}$
EventPeriod	: [. . .]	$EventPeriod[e] \in \{1, \ldots, Periods\}$
EventRoom	: [. . .]	$EventRoom[e] \in \{1, \ldots, Rooms\}$

Avec MiniZinc

H1

```
forall(e in 1..Events, r in 1..Rooms
        where EventRoomConstraints[e, r] = -1)
   (EventRoom[e] != r);
```

H5

```
forall(e in 1..Events, p in 1..Periods
        where EventPeriodConstraints[e, p] = -1)
   (EventPeriod[e] != p);
```

S2

```
PeriodPreferenceCost = sum(e in 1..Events)
      (EventPeriodConstraints[e, EventPeriod[e]]);
```

Avec toulbar2

Pour chaque **Event** e,
créer **une fonction de coût** portant sur la variable
EventRoom[e] : {1, … ,**Rooms**} → 0, …, **T**
r ↦ cost(r)

tq : Si **EventRoomConstraints**[e][r] vaut -1
alors cost(r) = **T**
sinon cost(r) = 0

Pour chaque **Event** e,
créer **une fonction de coût** portant sur la variable
EventPeriod[e] : {1, … ,**Periods**} → 0, …, **T**
p ↦ cost(p)

tq :

Si **EventPeriodConstraints**[e][p] vaut -1
alors cost(p) = **T**

sinon cost(p) = **EventPeriodConstraints**[e][p]

Nos travaux :

1) Programmer sous toulbar2 le problème d'emploi du temps.

2) Vérifier sa conformité au modèle MiniZinc (cohérences entre les réponses de toulbar2 et de MiniZinc).

3) Mesures de performances et comparaisons, améliorations

Ce dont on dispose :

- Code du modèle MiniZinc :

  exam_tt.mzn et **40** fichiers **data.dzn**

- Les données : une collection de 40 instances issues de 7 départements de **6 universités**

  → diversité de situations réelles.

- Comparaison de résultats (SA et MiniZinc)

Table 1. Main features of the instances.

Dept.	#inst	Courses		Events		Periods		Rooms		Slots	
		min	max	min	max	min	max	min	max	min	max
D1	7	239	281	239	281	26	52	64	65	2	2
D2	3	57	58	61	62	156	204	0	0	6	6
D3	9	76	89	78	177	48	188	14	15	4	4
D4	6	223	240	235	514	38	88	34	34	2	2
D5	5	125	156	132	426	24	136	17	20	2	2
D6	8	189	207	346	539	52	90	29	29	2	2
D7	2	60	63	136	150	155	330	22	22	5	10

Ce dont on dispose :

Table 3. Comparative results.

Instance	SA (Simulated Annealing / Recuit Simulé)			MiniZinc				toulbar2
	avg.	best	time	chuffed	gecode	coin-bc	cplex	
D1-1-16	**180.40**	180	568.1	×	—	×	×	
D1-1-17	**134.00**	134	499.2	×	—	×	×	
D1-2-16	**258.63**	257	541.6	×	—	×	×	
D1-2-17	**352.00**	351	698.9	×	—	×	×	
D1-3-16	**478.37**	477	483.3	×	—	×	×	
D1-3-17	**354.57**	354	493.1	×	—	×	×	
D1-3-18	**80.00**	80	536.6	×	—	×	×	
D2-1-18	427.77	426	94.7	—	8731	906	**406**	*426...*
D2-2-18	22.00	22	88.7	1543	4022	**0**	**0**	*22*
D2-3-18	22.00	22	95.0	1873	3985	**0**	**0**	*22*
D3-1-16	**0.00**	0	61.9	—	75947	×	—	*0*
D3-1-17	**0.00**	0	83.5	—	82948	×	—	*0*
D3-1-18	**0.00**	0	83.9	—	82433	×	—	*0*
D3-2-16	**0.00**	0	0.8	**0**	—	—	**0**	*0*
D3-2-17	**0.00**	0	3.1	**0**	—	—	**0**	*0*
D3-2-18	**0.00**	0	3.5	—	—	—	**0**	*0*
D3-3-16	**0.00**	0	1.0	**0**	—	—	**0**	*0*
D3-3-17	**0.00**	0	2.3	**0**	—	—	**0**	*0*
D3-3-18	**0.00**	0	2.2	**0**	—	—	**0**	*0*
D4-1-17	**132.43**	18	312.0	×	—	×	×	
D4-1-18	**567.73**	563	1401.4	×	—	×	×	
D4-2-17	**575.50**	566	1307.3	×	—	×	×	
D4-2-18	**11609.33**	9685	39.3	×	—	×	×	

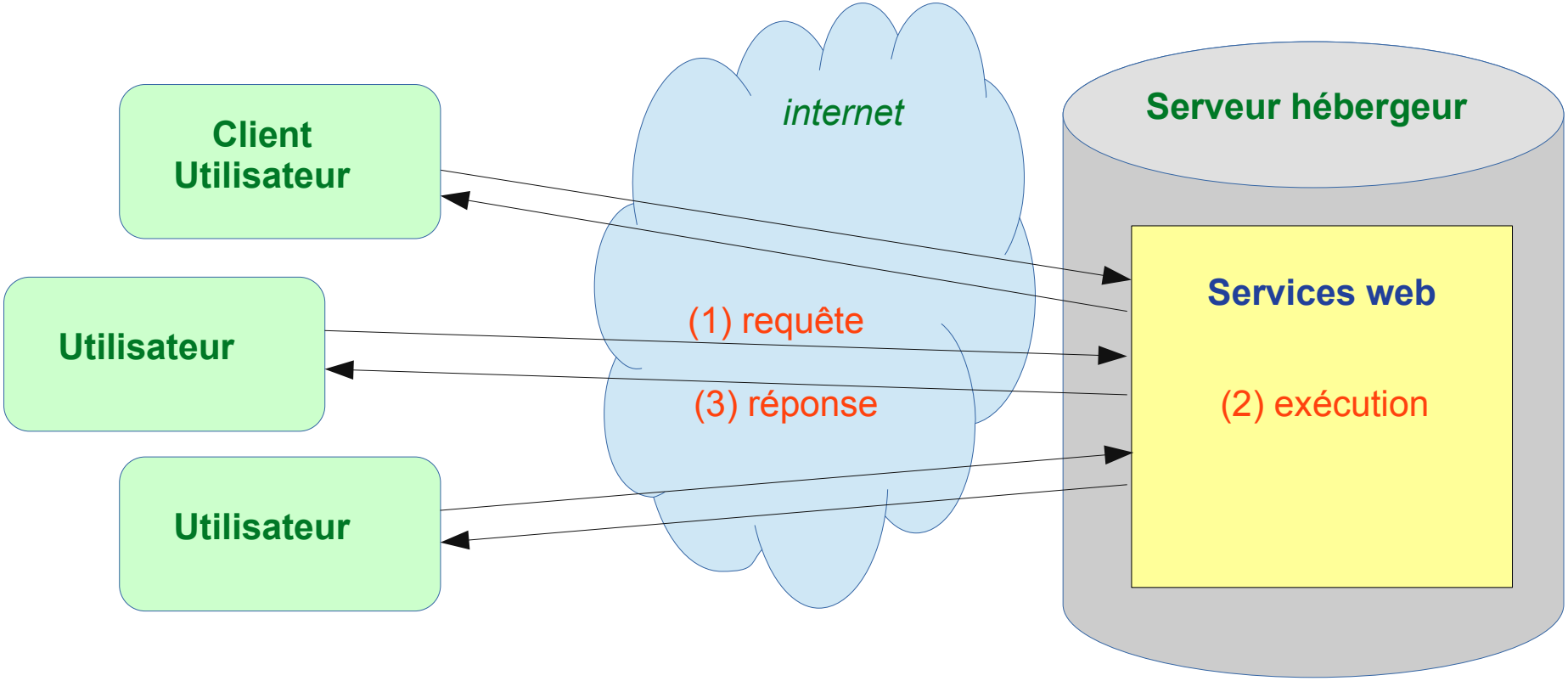
x : *the solver exhausted the memory.*

— : *the solver has not been able to produce any feasible solution within the time limit. (for minizinc, a timeout of 1h for each run)*

C. Mise en œuvre, utilisation de services web

Services web

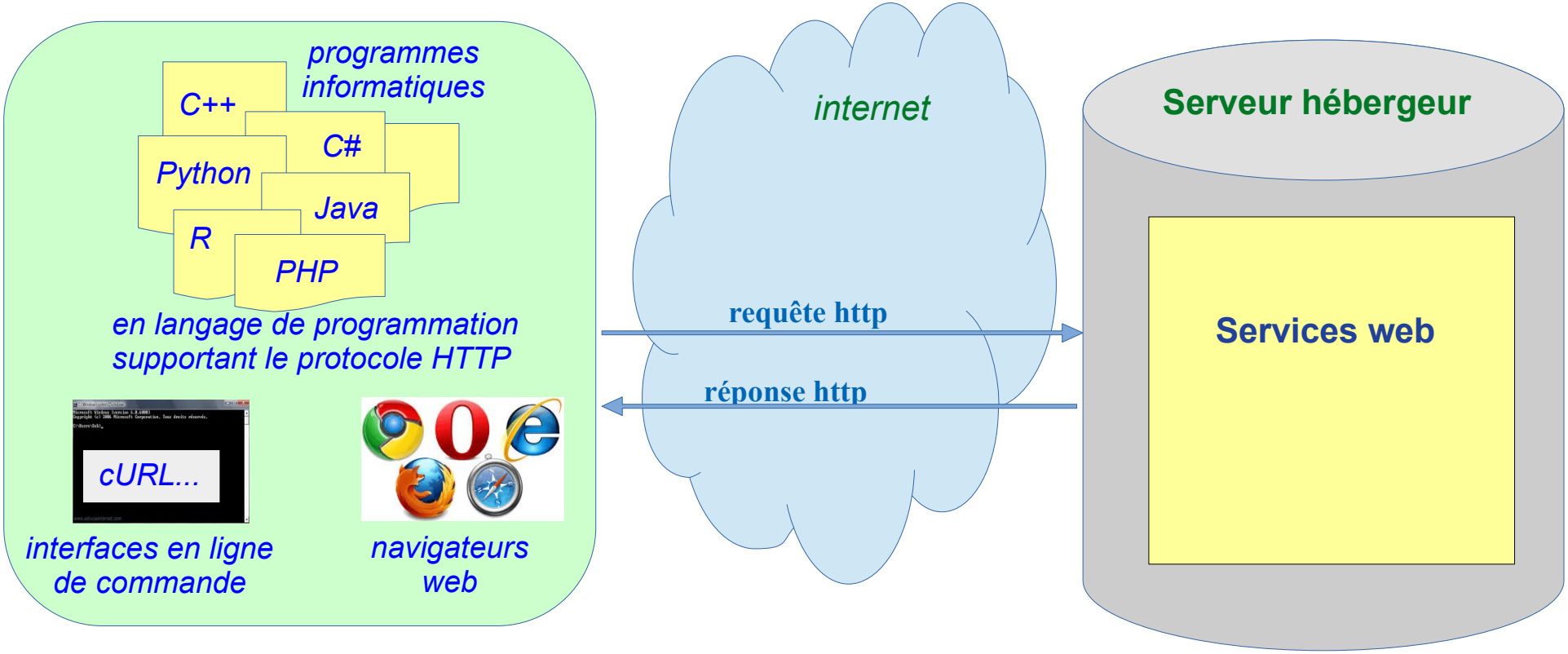
Services web : logiciel s'exécutant sur un serveur accessible depuis **internet** et répondant à des demandes, appelées **requêtes**.



Services web

Services web : logiciel s'exécutant sur un serveur accessible depuis **internet** et répondant à des demandes, appelées **requêtes**.

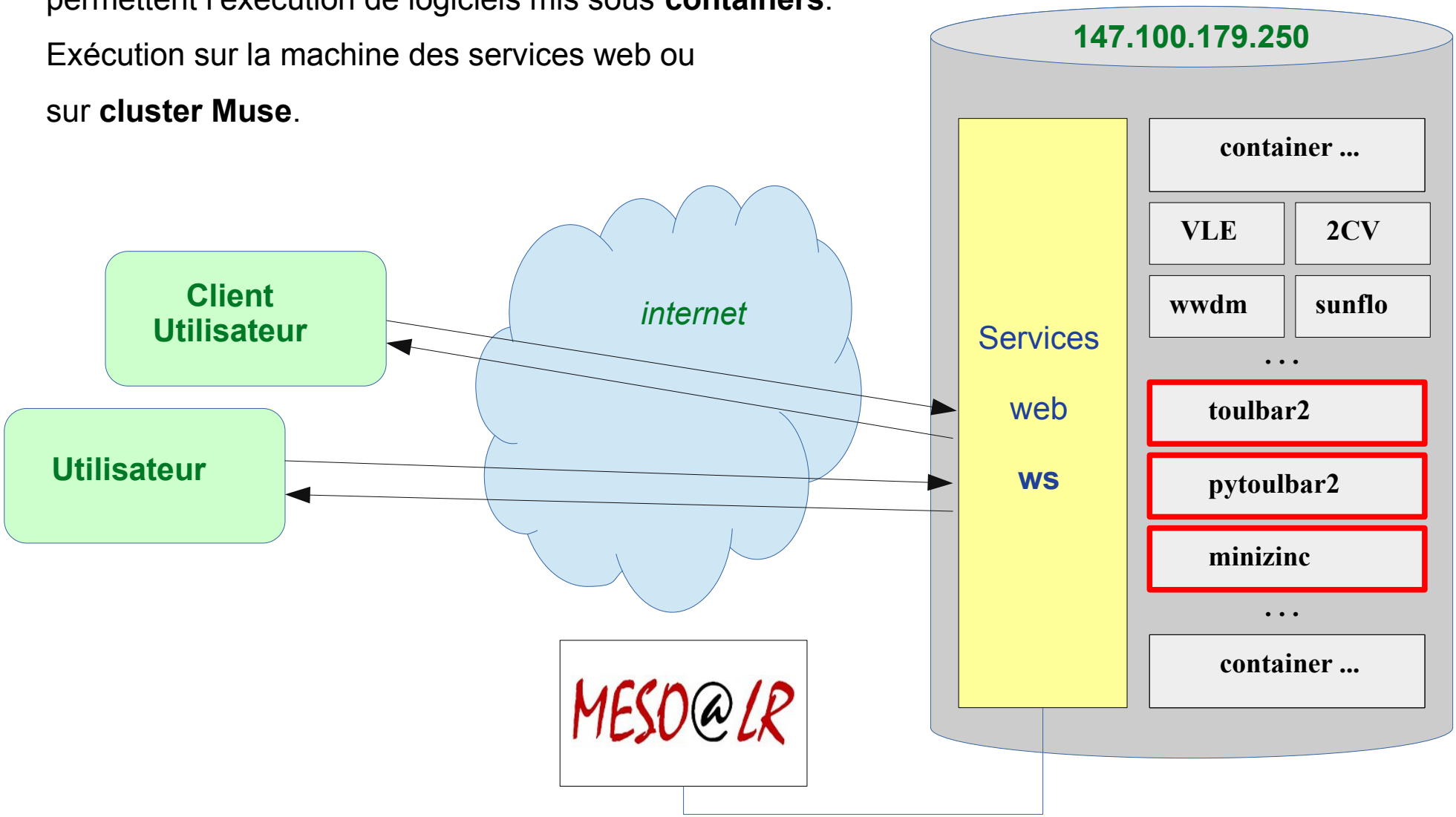
Les services web basés sur le protocole **HTTP** peuvent être appelés depuis tout 'objet' capable d'envoyer et de recevoir des **requêtes HTTP**.



Démonstrateur de services web ws

Les **services web ws**, développés dans l'équipe **RECORD** (démonstrateur), permettent l'exécution de logiciels mis sous **containers**.

Exécution sur la machine des services web ou sur **cluster Muse**.



Démonstrateur de services web ws → éléments informatiques

Logiciel des services web ws développé avec :

- **Python 3.7**
- Framework de développement web **django** (djangoproject.com)
- **django-rest-framework** (django-rest-framework.org , application django)
- **SQLite** (database)
- **Sphinx** (documentation)

Déploiement :

- sur une **Machine Virtuelle** fournie par la **DSI INRAE** :
  machine sous **Debian 10** (Buster), serveur **Apache2**.
- HPC sur **Cluster Muse meso@LR** :
  Mésocentre de calcul de Montpellier (https://meso-lr.umontpellier.fr)

URL services web ws : http://147.100.179.250

Démonstrateur de services web ws

URLs des requêtes principales :

- Liste des logiciels disponibles :

  http://147.100.179.250/api/**software/list**/
- Détails sur un de ces logiciels :

  par exemple http://147.100.179.250/api/**software/detail/pytoulbar2**/
- Demande d'exécution (sur machine des services web) :

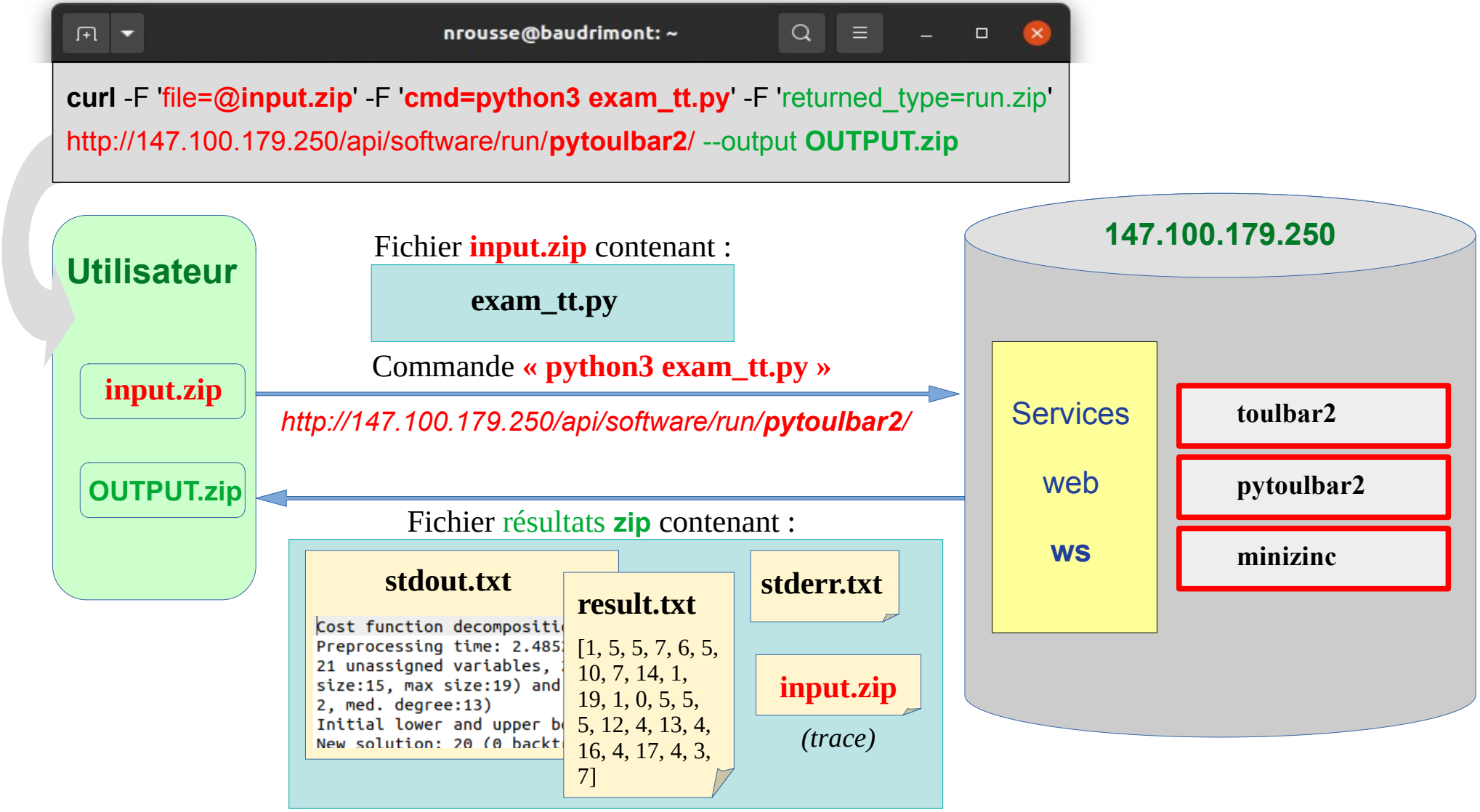
  par exemple http://147.100.179.250/api/**software/run/pytoulbar2**/
- Demande d'exécution sur cluster Muse :

  par exemple http://147.100.179.250/api/**software/muse/run/pytoulbar2**/

Authentification requise pour les demandes d'exécution (*http://147.100.179.250/.../run/.../*)

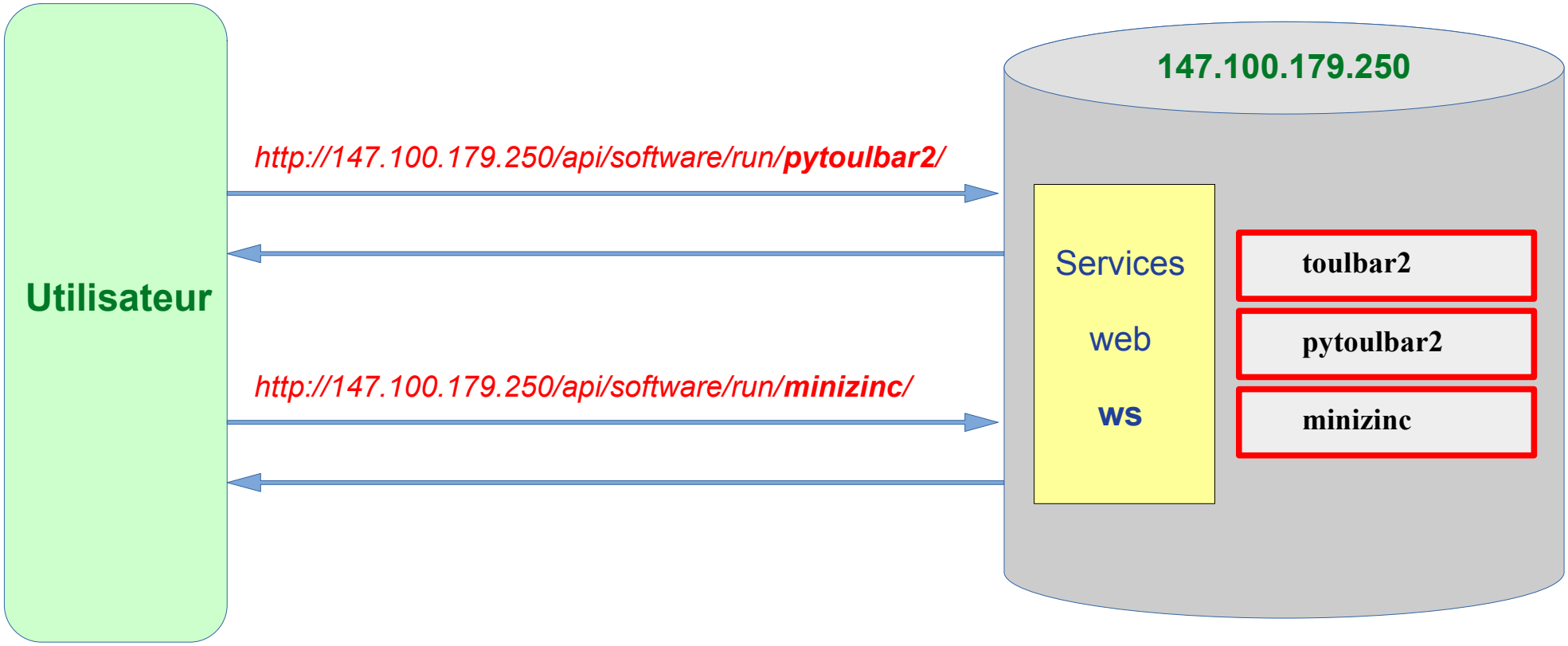
Utilisation des services web ws pour le problème d'emploi du temps

- **toulbar2**

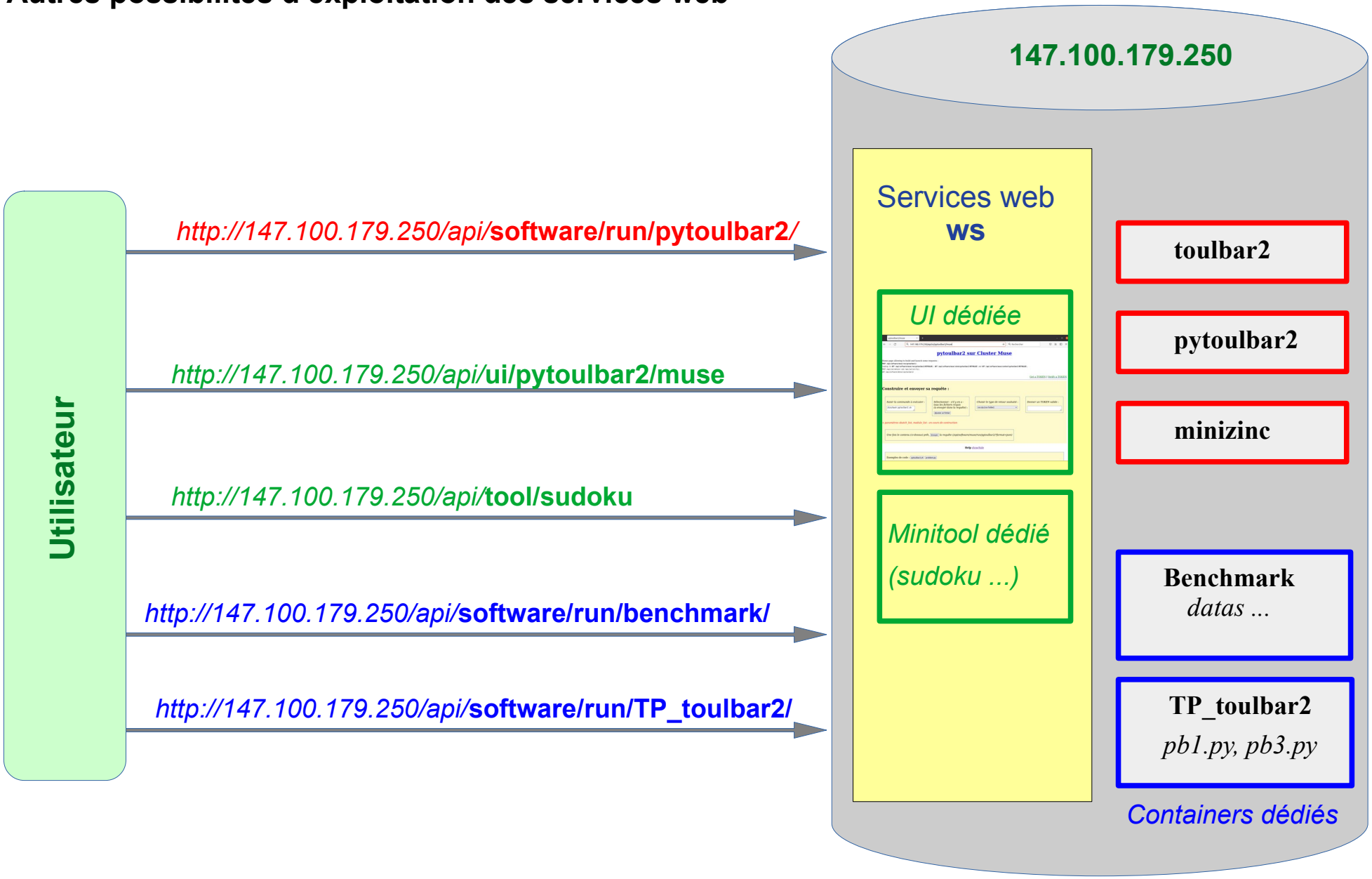


Utilisation des services web pour le problème d'emploi du temps

- **toulbar2**
- **minizinc**



Autres possibilités d'exploitation des services web



User Interface dédiée facilitant l'appel des services web

Exemple d'une page web de construction et envoi d'une requête demandant d'exécuter « python3 problem.py »

pytoulbar2/muse

147.100.179.250/api/ui/pytoulbar2/muse

Rechercher

pytoulbar2 sur Cluster Muse

Home page allowing to build and launch some requests :

POST /api/software/muse/run/pytoulbar2/,

leading to **GET /api/software/muse/run/pytoulbar2/KEYVALUE/**, **GET /api/software/muse/state/pytoulbar2/KEYVALUE/** *and* **GET /api/software/muse/content/pytoulbar2/KEYVALUE/**,

POST /api/jwt/obtain/ and /api/jwt/verify/,

GET /api/software/detail/pytoulbar2/

Get a TOKEN | Verify a TOKEN

Construire et envoyer sa requête :

Saisir la commande à exécuter :

/bin/bash pytoulbar2.sh

Sélectionner - s'il y en a - tous les fichiers requis (à envoyer dans la requête) :

Ajouter un fichier

Choisir le type de retour souhaité :

run.zip (run folder)

Donner un TOKEN valide :

+ paramètres sbatch_list, module_list : en cours de contruction

Une fois le contenu (ci-dessus) prêt, Envoyer *la requête (/api/software/muse/run/pytoulbar2/?format=json)*

Help *show/hide*

Exemples de code : pytoulbar2.sh problem.py

pytoulbar2.sh intervient dans `Saisir la commande à exécuter:/bin/bash pytoulbar2.sh`. En l'état, ce script pytoulbar2.sh appelle **problem.py** (`python3 problem.py`) qu'il faudra aussi envoyer dans la requête avec pytoulbar2.sh.

Vous pouvez **adapter** le code de pytoulbar2.sh et problem.py à votre propre cas.

Il faut **enregistrer** pytoulbar2.sh et problem.py avant de pouvoir les envoyer dans une requête.

Minitool en accès public

Exemple d'une requête dédiée à un outil 'sudoku' qui a été programmé dans les services web, permettant de choisir (parmi 18000) une grille de sudoku, de la résoudre et d'en recevoir la solution.

```
nrousse@baudrimont: ~
nrousse@baudrimont:~$ curl -F 'returned_type=stdout.txt'
-F 'indice=4455' http://147.100.179.250/api/tool/sudoku
sys.argv :  ['sudoku.py', '4455']
INDICE :  4455

Solution found :
7 3 8 6 9 1 2 4 5
6 4 2 5 8 7 1 9 3
5 1 9 3 2 4 8 6 7
2 5 4 7 6 9 3 1 8
8 7 1 4 3 5 9 2 6
3 9 6 2 1 8 5 7 4
4 6 3 1 5 2 7 8 9
9 2 7 8 4 3 6 5 1
1 8 5 9 7 6 4 3 2

for grid :
0 0 0 0 0 0 2 4 5
6 0 0 0 8 0 0 0 0
5 1 0 0 0 0 8 6 0
0 0 0 7 0 9 3 0 8
8 0 0 4 0 0 0 2 6
0 0 0 2 1 0 0 0 0
0 0 0 0 5 0 7 8 9
0 2 7 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

nrousse@baudrimont:~$
```

Pour ce genre d'outil, l'exécution est permise sans s'authentifier => le paramétrage de la requête est fortement bridé.

```
nrousse@baudrimont: ~
curl -F 'returned_type=stdout.txt' -F 'indice=4455' http://147.100.179.250/api/tool/sudoku
```

Paramètre 'indice' : indice de la grille à résoudre, valeur dans 0..17999.