# Compositions de services: réduction de graphe avec variabilité des QoS

Yanik NGOKO

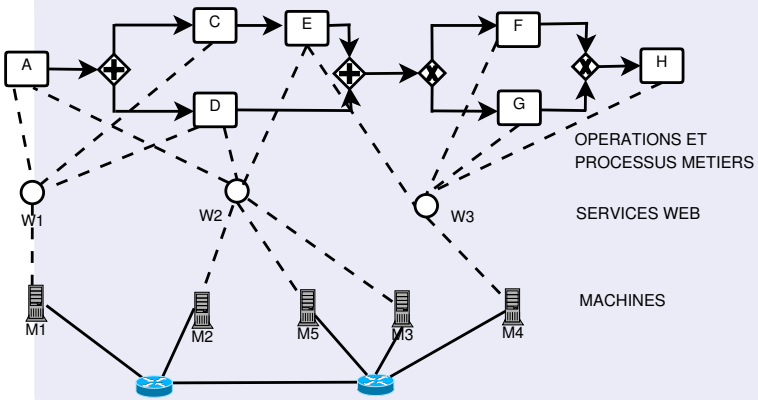
University of Paris 13, France

Vichy, June 3 2014

# Introduction

## Intérêts

- Orchestration et choreographies de services.
- Composition dynamique des services.
- Applications à la gouvernance électronique et aux villes intelligentes.




- Focus sur le contexte cloud
- Algorithme de prédiction rapide
- Capturer la variabilité des QoS

# Types de QoS

## QoS déterministes

- Pour une opération $A$ (login, écriture, recherche etc.), on a son temps de réponse $t_A \in \mathbb{R}^+$.
- Aucune prise en compte de la variabilité.

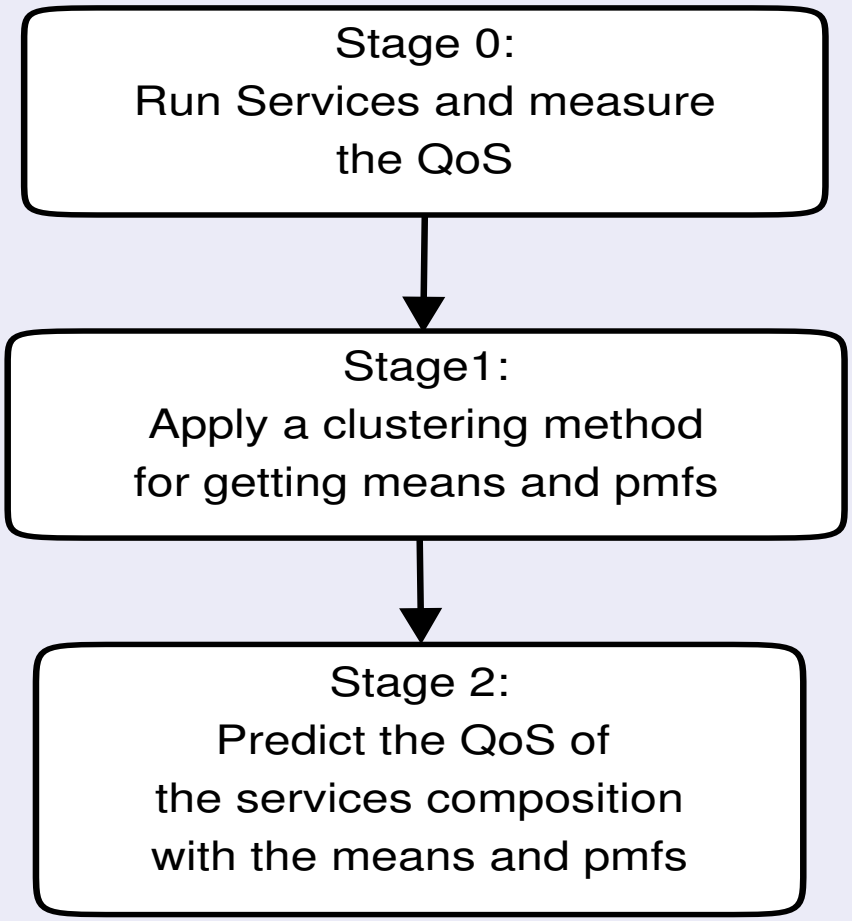
## QoS stochastiques

- Pour une opération $A$, on a la loi de distribution du temps de réponse $t_A$
- Comment utiliser la loi dans un processus calculable?

## QoS probabilistes

- Pour une opération $A$, on un tuple $(A, f_A)$. $A$ est un ensemble fini de valeurs réelles pour le temps de réponse et $f_A$ sont les probabilités masses fonction pour ces valeurs. Exemple: $A = \{0.4, 5, 8.2, 12\}$, $f_A = \{0.2, 0.3, 0.3, 0.4\}$

# QoS probabilistes

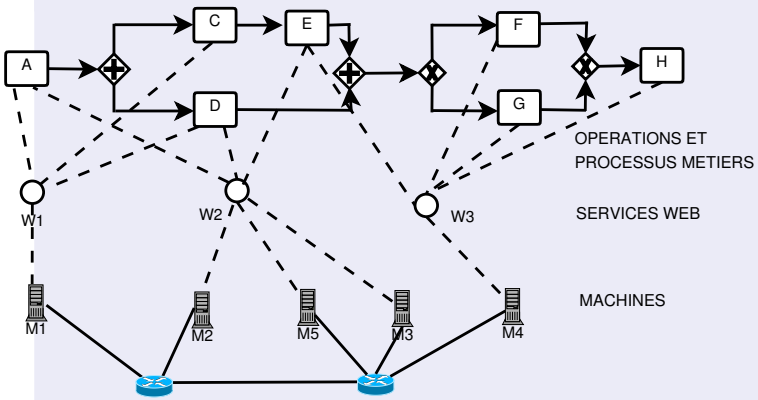



Remarques: Etant donné $(X, f_X)$, on a $|f_X| = |X|$ et $\sum_{v \in X} f_X(v) = 1$

# Prédire les QoS d'une composition

## But

Estimer les QoS d'une collaboration de services à partir de celle des opérations.



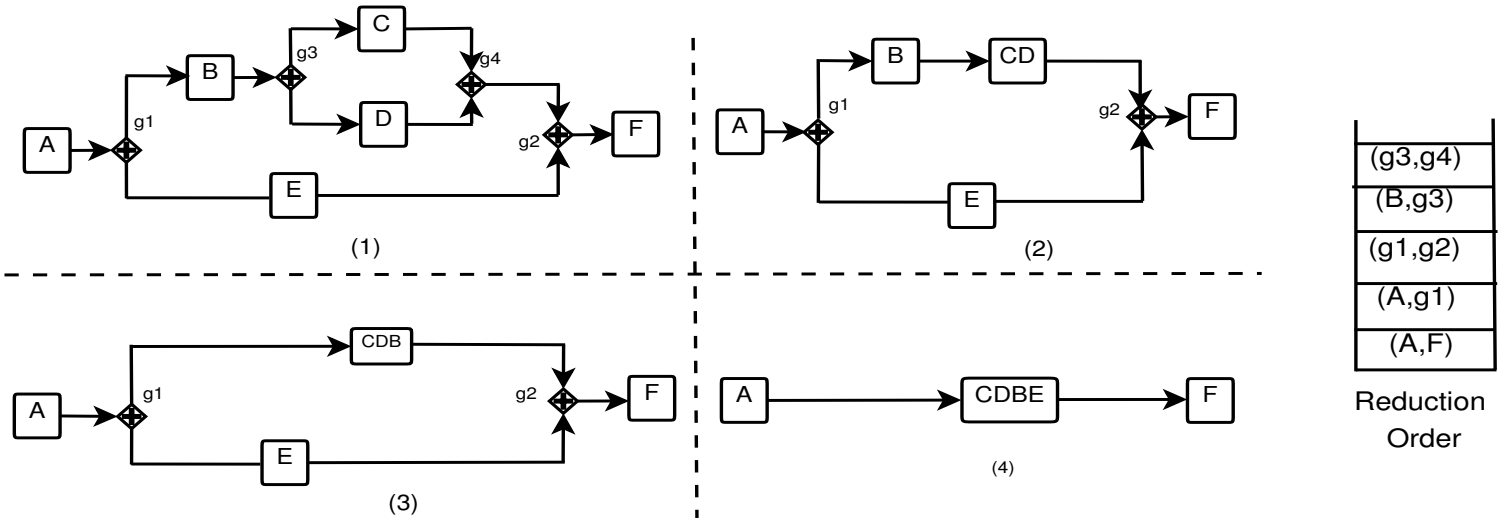

Focus: réduction des graphes.

- Réduction des graphes
- Programmation linéaire
- algèbre des processus et compilation

# Concepts dans la réduction

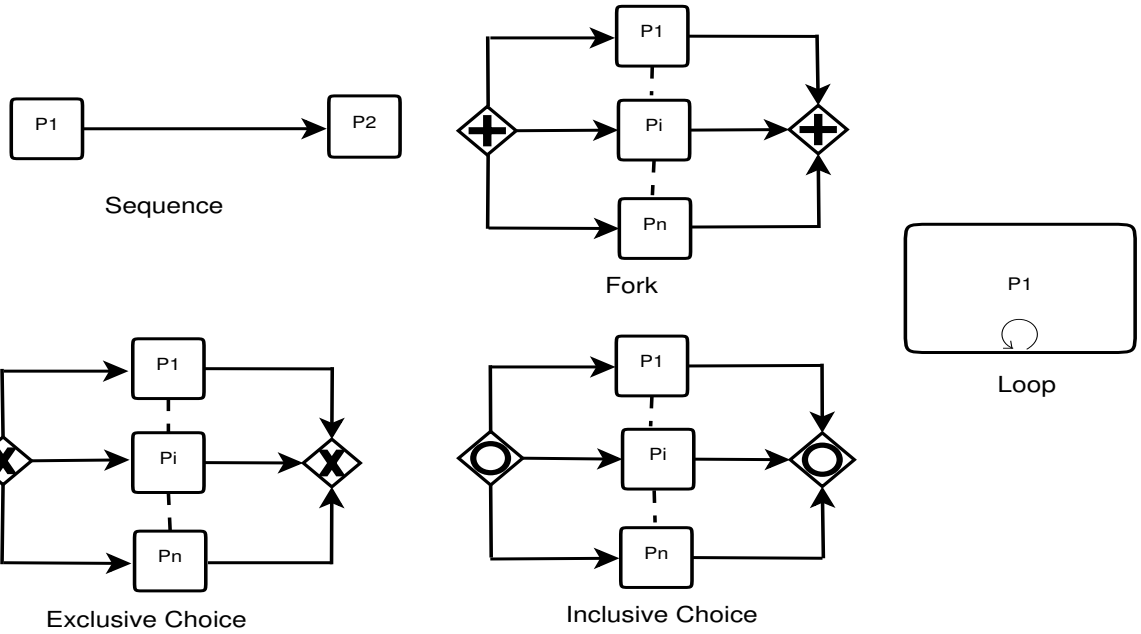
| Notions | Description of an instance |
|---|---|
| Graph reduction algorithm | Algorithm for computing the QoS of a graph by progressively substituting a subgraph pattern by a single node |
| Aggregation rules | Semantics rules stating how to aggregate QoS on some subgraph patterns: sequence, fork/join, split/join subgraphs etc. |
| QoS of operations | Response time, availability, price, energy consumption etc. |

# Exemple


A
B
C
D
E
F
g1
g2
g3
g4
(1)
A
B
CD
E
F
g1
g2
(2)
A
CDB
E
F
g1
g2
(3)
A
CDBE
F
(4)
(g3,g4)
(B,g3)
(g1,g2)
(A,g1)
(A,F)
Reduction
Order

# Patterns et règles d'aggrégation

Hypothèses: Graphes structurés.




| **Sequence** | **Fork** | **Loop** |
|---|---|---|
| $T(P_1)+T(P_2)$ | $\max\{T(P_1),\ldots T(P_n)\}$ | $m.T(P_1)$ |
| **Exclusive choice** | **Inclusive choice** | |
| $\sum_{i=1}^{n} p_i.T(P_i)$ | $p_{or1}.T(P_1)+p_{or2}.T(P_2) + p_{or\|\|}.\max\{T(P_1),T(P_2)\}$ | |

# Redéfinir les règles d'aggrégation (I)

| **Sequence** | **Fork** | **Loop** |
|---|---|---|
| $T(P_1)+T(P_2)$ | $\max\{T(P_1),\dots T(P_n)\}$ | $m.T(P_1)$ |
| **Exclusive choice** | **Inclusive choice** | |
| $\sum_{i=1}^{n} p_i.T(P_i)$ | $p_{or1}.T(P_1)+p_{or2}.T(P_2) \ +p_{or\|\|}.\max\{T(P_1),T(P_2)\}$ | |

## Observation

Les régles d'aggrégation sont données pour le cas déterministe. Il faut les redéfinir pour le cas probabiliste: que vaut $T(P_1) + T(P_2)$ si $T(P_1) = [(1, 5), (0.1, 0.9)]$ et $T(P_2) = [(0.3, 0.5), (0.3, 0.7)]$?
On peut se contenter dans la redéfinition d'un ensemble d'*opérations noyaux*.

# Redéfinir les règles d'aggrégation (II)

**Addition:** $(X, f_X) + (Y, f_Y) = (Z, f_Z)$. Chaque $z_i \in Z$ est la somme d'une valeur $x_a \in X$ et $y_b \in Y$; sa pmf est

$$f_Z(z_i) = \sum_{x_a + y_b = z_i} f_X(x_a).f_Y(y_b)$$

**Multiplication scalaire:** A tuple $\alpha.(X, f_X) = (Z, f_Z)$; $f_Z = f_X$ et $Z = \alpha.X$

**Maximum:** $\max\{(X, f_X), (Y, f_Y)\} = (Z, f_Z)$: $Z = X \cup Y$ et on a les pmfs suivantes:

$$f_Z(z_i) = \begin{cases} f_X(z_i). \sum_{\substack{y_b < z_i \\ y_b \in Y}} f_Y(y_b) \text{ si } \begin{smallmatrix} z_i \in X \\ z_i \notin Y \end{smallmatrix} \\ f_Y(z_i). \sum_{\substack{x_a < z_i \\ x_a \in X}} f_X(y_b) \text{ si } \begin{smallmatrix} z_i \notin X \\ z_i \in Y \end{smallmatrix} \\ f_X(z_i). \sum_{\substack{y_b \leq z_i \\ y_b \in Y}} f_Y(y_b) + f_Y(z_i). \sum_{\substack{x_a \leq z_i \\ x_a \in X}} f_X(x_a) \text{ sinon} \end{cases}$$

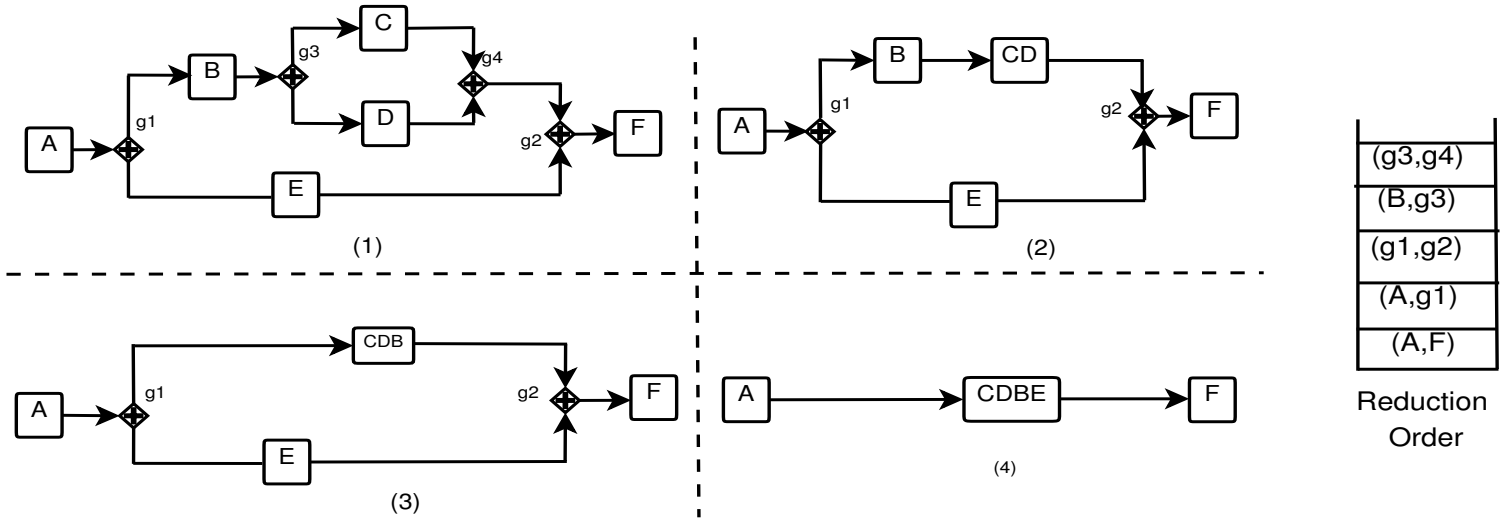
# Exemple

$$T(P_1) = [(1,5),(0.1,0.9)] \text{ et } T(P_2) = [(0.3,0.5),(0.3,0.7)]?$$

$$T(P_1) + T(P_2) = [(1.3,1.5,5.3,5.5),(0.03,0.07,0.27,0.63)]$$

$$0.3.T(P_1) = [(0.3,1.5),(0.1,0.9)]$$

# Explosion combinatoire dans l'algorithme de réduction




L'addition de deux QoS $(X, f_X) + (Y, f_Y) = (Z, f_Z)$ donne un ensemble $Z$ tel que $|Z| \geq \max\{|X|, |Y|\}$

# En théorie

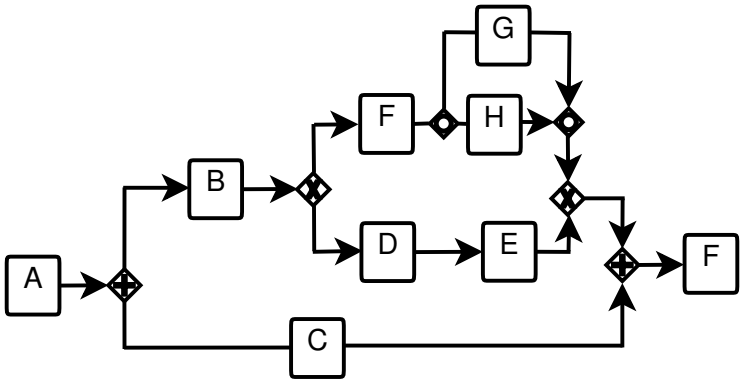
## Lemma

*Let us consider n operations $x_1, \ldots, x_n$ of an elementary sequence subgraph. Each operation $x_i$ has a response time $(X_i, f_{X_i})$. There exists a distribution of the operation response times for which the reduction of the sequence will result in an operation node z whose precision size is $|Z| = |X_1|.|X_2|.\ldots.|X_n|$.*

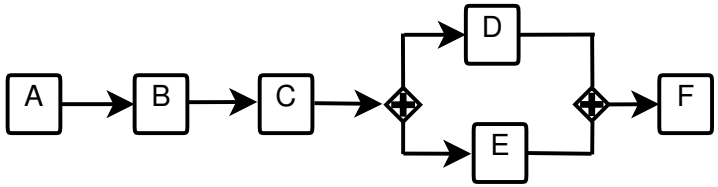
## Theorem

*(1) The reduction algorithm cannot always be executed in a computer with finite memory. (2) The reduction of an elementary subgraph can require an exponential number of arithmetical operations (in the number of subgraph nodes). (3) For a fixed n, there exists an infinite number of elementary subgraphs with n operations nodes $x_1, \ldots, x_n$ such that the precision size of the operation z, obtained from their reduction, is $|Z| = |X_1|.|X_2|.\ldots.|X_n|$.*

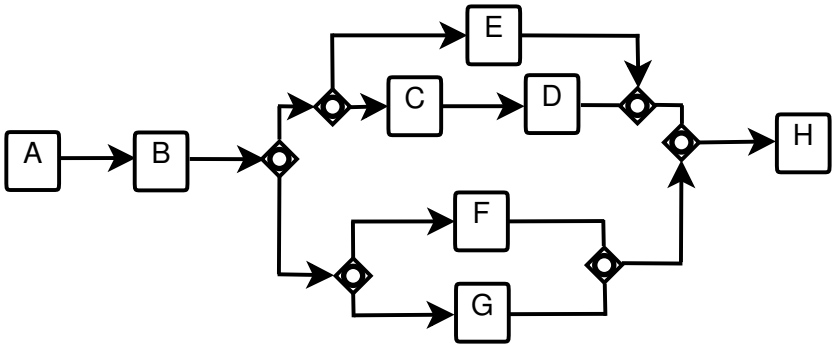
# En pratique



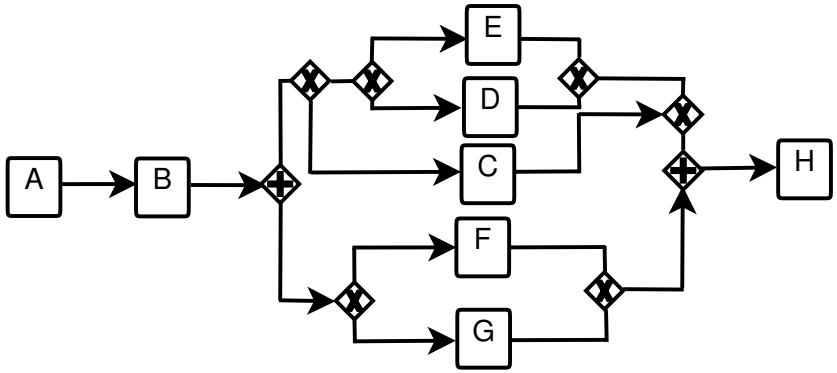

(a) Shipment




(b) Procurement



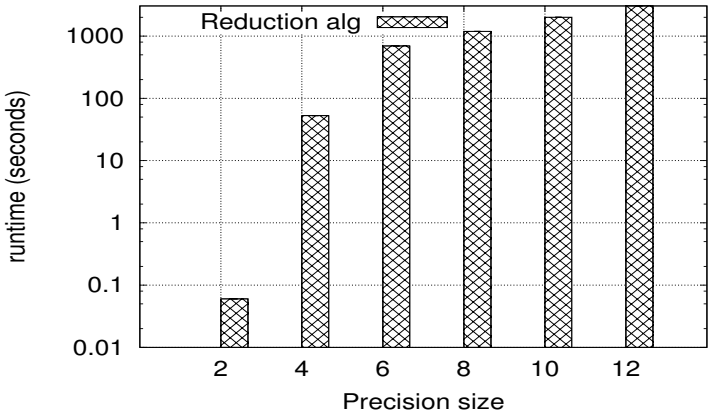

(c) Disbursement



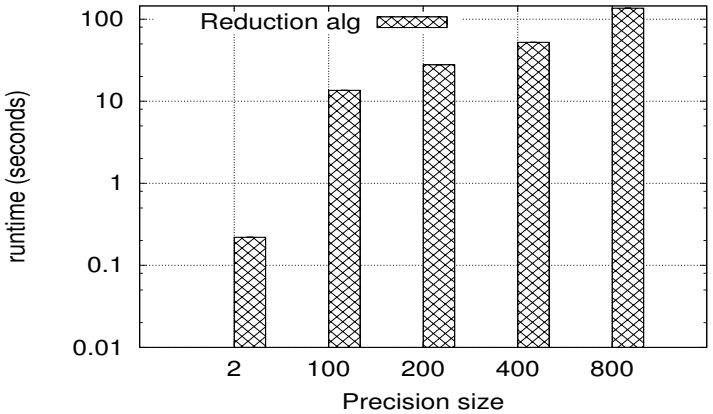

(d) Meal Options

Figure : Exemples de processus

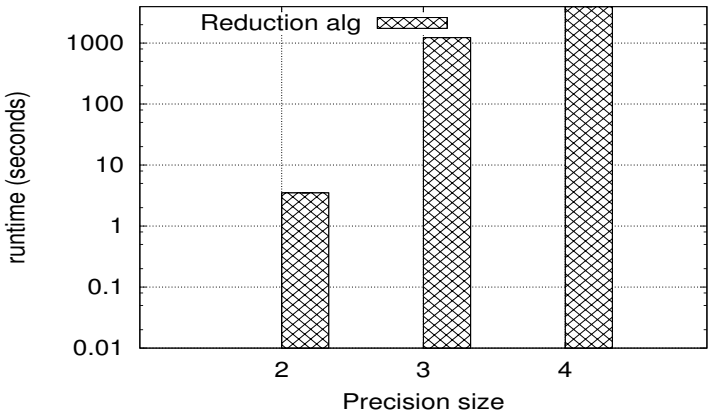
# En pratique



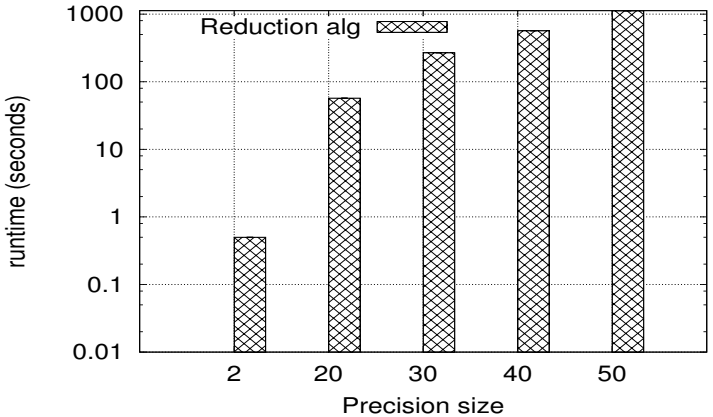

(a) Shipment




(b) Procurement




(c) Disbursement




(d) Meal Options

# Gestion de l'explosion combinatoire

## Observations

- L'arithmétique des réels a des similarités: précision du calcul à priori non bornée sur les réels
- L'addition exacte de deux nombres flottants peut entrainer un accroissement de la prédiction du résulat: comment cela est-il géré?

## Principes

- Principe du régistre: fixer une précision maximale dans tout calcul. Sur le résultat $(Z, f_Z)$ on s'assure que $|Z| \leq k$ ($k$ fixé).
- Principe de la normalisation: Si $(Z, f_Z)$ est le résultat d'un calcul, le normaliser à $(Z', f_{Z'})$ tel que $|Z'| \leq k$ et $(Z', f_{Z'}) \sim_k (Z, f_Z)$. La relation de similitude $(\sim_k)$ est à définir.

# Algorithme de réduction revisité

Build a reduction order $ORD$ from $G_o$.
**while** $ORD$ is not empty **do**
Unstack a subgraph $Sbg$ from $G_o$.
Aggregate the service response time of $Sbg$ and save the result in $(Z, f_Z)$;
Replace the nodes of $Sbg$ in $G_o$ by an operation node whose response time is $(Z, f_Z)$.
**if** $|Z| > k$ **then**
Replace $(Z, f_Z)$ by a normalized tuple $(Z', f_{Z'})$.

# Comment définir la similitude?

## Deux approches inspirées du calcul des flottants

- Troncature: Etant donné $(Z, f_Z)$, on supprime certains éléments de $Z$ et on met à jour $f_Z$
- Arrondi: On arrondi des éléments de $Z$ à une valeur unique et on met à jour $f_Z$

# Définitions formelles: Troncature

$(Z', f_{Z'}) \sim_k (Z, f_Z)$ si:

1. $$Z' \subset Z$$
2. For each value $z'_j \in Z'$, we define the set $Cl_j = \{z_u \in Z \text{ s.t } j = \arg\min_{1 \leq l \leq k} |z_u - z'_l|^2\}$. The pmf in the normalized tuple is defined by the equation
$$f_{Z'}(z'_i) = \sum_{y \in Cl(z'_i)} f_Z(y)$$
3. The choice of $Z'$ must minimize
$$E = \sum_{i=1}^{k} \sum_{z_u \in Cl_i} |z_u - z'_i|^2$$

Ici, $E$ quantifie la perte d'information.

# Exemple: Troncature

Etant donné $(Z, f_Z) = [(0.2, 0.8, 0.36, 0.6), (0.4, 0.3, 0.2, 0.1)]$
trouver $(Z', f_{Z'}) \sim_2 (Z, f_Z)$

$$(Z', f_{Z'}) = [(0.36, 0.6), (0.6, 0.4)]$$

# Algorithme pour le choix de $(Z', f_{Z'})$: cas de la troncature

1: $t = 1$;
2: Arbitrarily choose $k$ values of $Z'(t)$ from $Z$;
3: Compute the clusters $Cl_j(t)$ such that $z_u \in Cl_j(t)$
4: if $j = \arg \min_{1 \leq l \leq k} |z_u - z'_l(t)|^2$;
5: $cerr = \sum_{i=1}^{k} \sum_{z_u \in Cl_i(t)} |z_u - z'_i(t)|^2$ ;
6: $oerr = +\infty$;
7: **while** $|oerr - cerr| > tol$ **do**
8: $t = t + 1$;
9: **for** $j = 1, \ldots, k$ **do**
10: Define $z'_j(t)$ as the median point of $Cl_j(t-1)$;
11: Build $Cl_j(t), j = 1, \ldots, k$;
12: $oerr = cerr$;
13: $cerr = \sum_{i=1}^{k} \sum_{z_u \in Cl_i(t)} |z_u - z'_i(t)|^2$;
14: Compute the values of $f_{Z'_j(t)}, j = 1, \ldots, k$;
15: Return $(Z'(t), f_{Z'(t)})$;

# Définitions formelles: Arrondi

$(Z', f_{Z'}) \sim_k (Z, f_Z)$ si:

1. $Cl_1, \ldots, Cl_k$ is a partition of $Z$ such that $Cl_j = \{z_u \in Z \text{ s.t } j = \arg \min_{1 \leq l \leq k} |z_u - z'_l|^2\}$; where

$$z'_j = \frac{1}{|Cl_j|} \sum_{y \in Cl_j} y$$

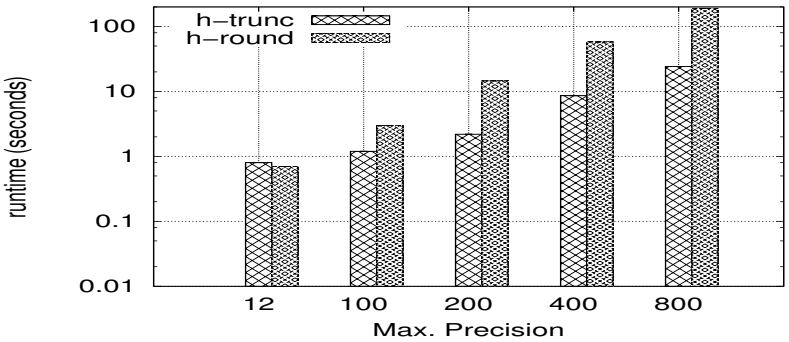
($Z'$ is created from the mean points of $Z$)

2. $$f_{Z'}(z'_i) = \sum_{y \in Cl(z'_i)} f_Z(y)$$

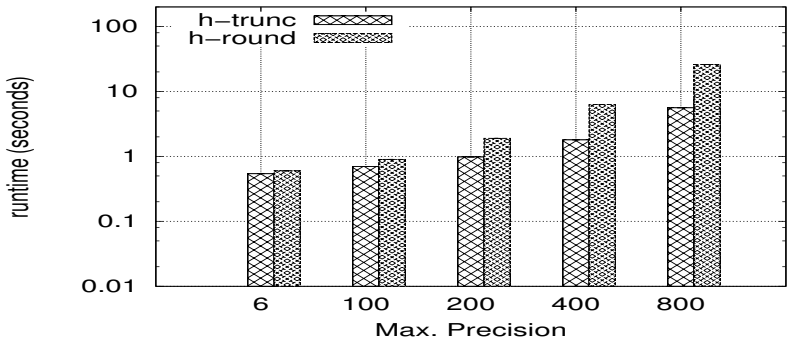
3. The choice of $Z'$ must minimize

$$E = \sum_{i=1}^{k} \sum_{z_u \in Cl_i} |z_u - z'_i|^2$$

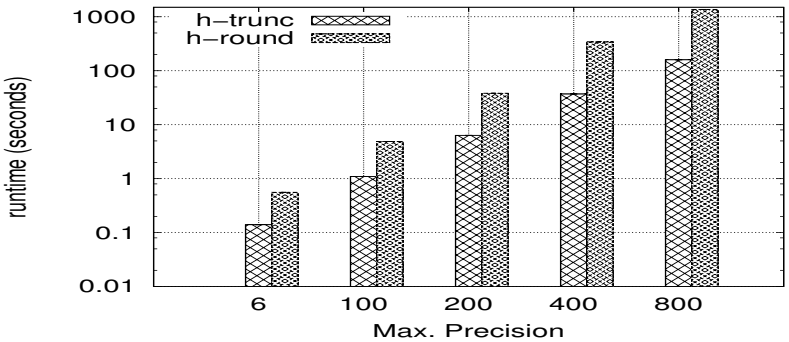
# Temps d'exécution



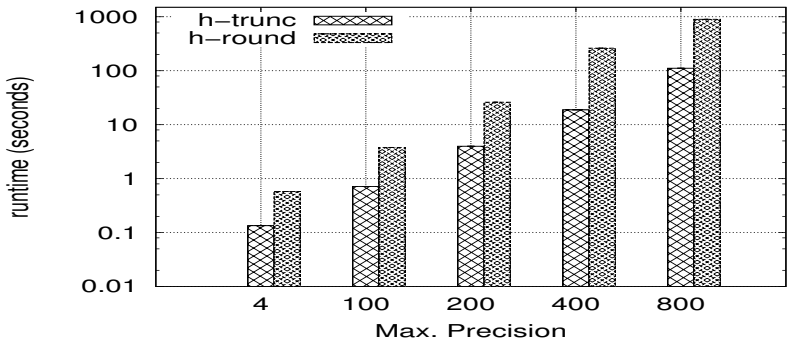

(a) Shipment, Initial precision = 12




(b) Shipment, Initial precision = 6




(c) Disbursement, Initial precision = 6

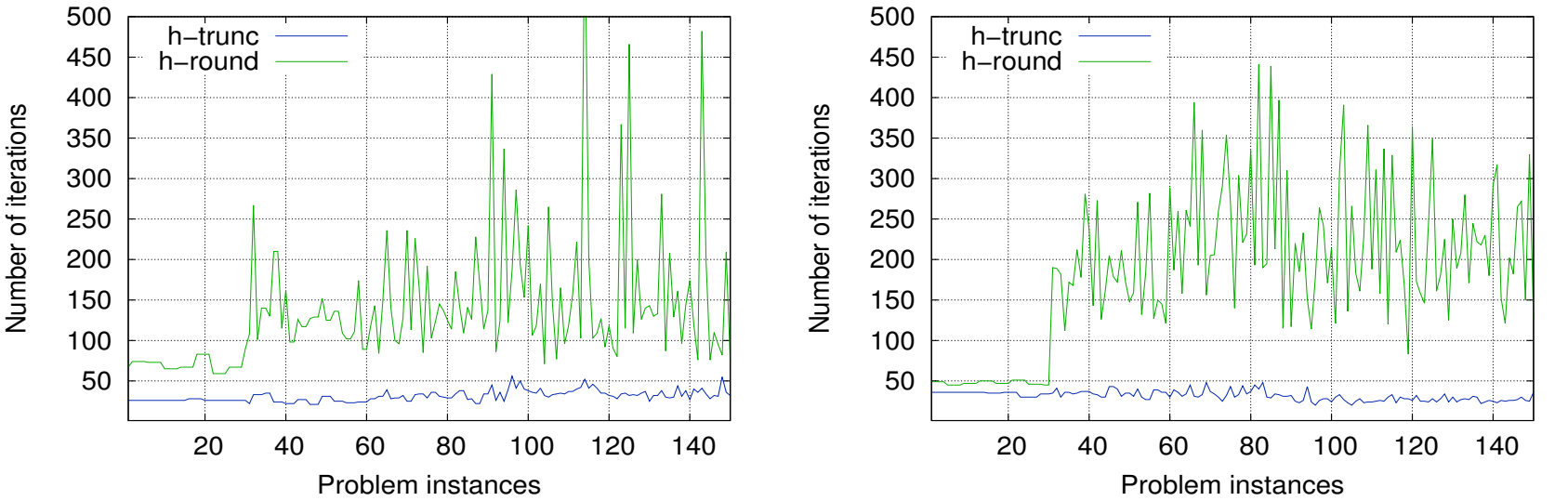



(d) Disbursement, Initial precision = 4

Figure : Runtime of *h-trunc* and *h-round* $tol = 1e^{-6}$

# Nombre d'itérations

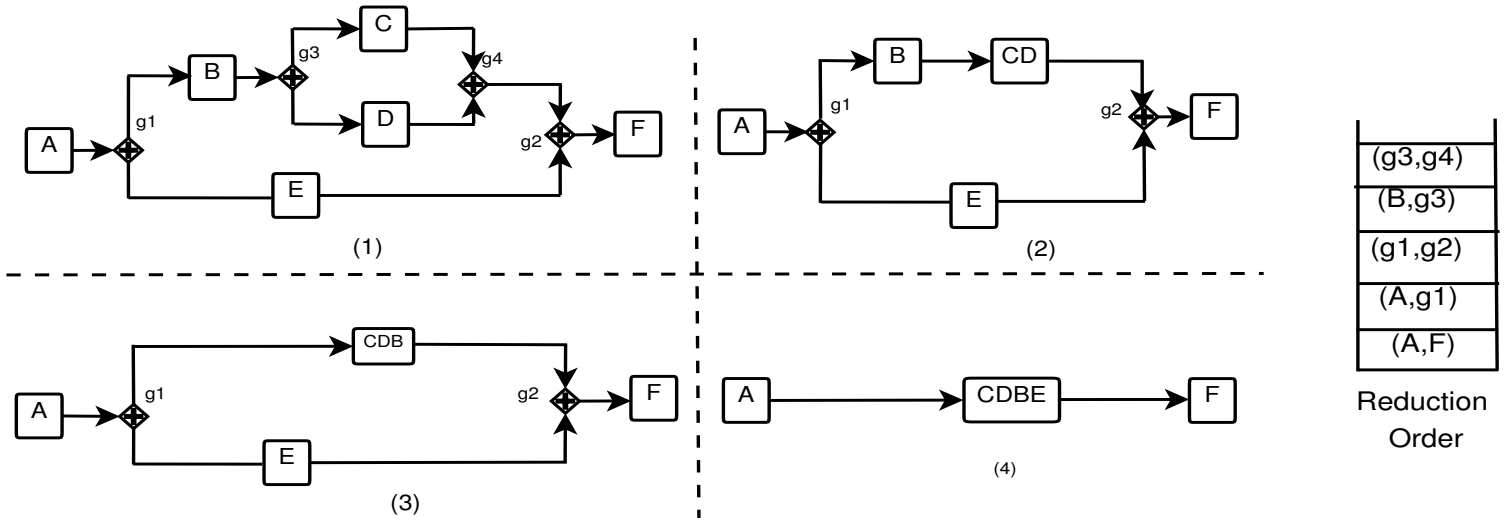
Loops on the condition $|oerr - cerr| > tol$.




(a) Shipment, Initial precision = 6 (b) Disbursement, Initial precision = 3

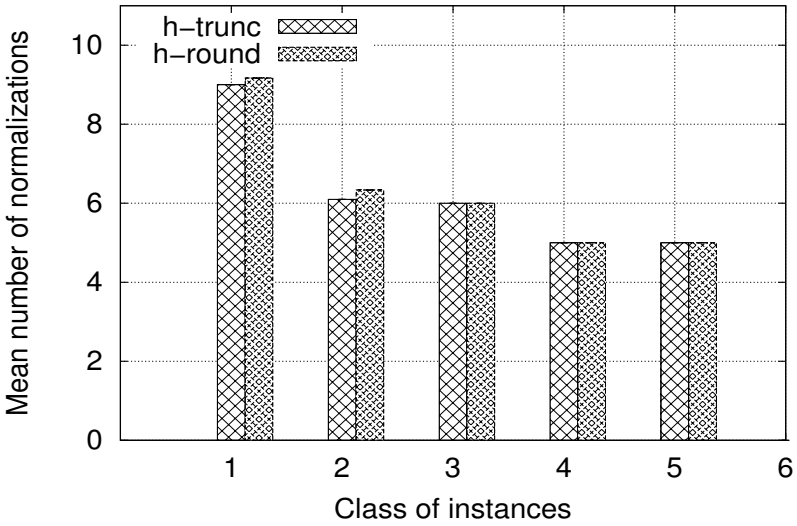
Figure : Number of iterations (on the condition $|oerr - cerr| > tol$) per problem instances. $tol = 1e^{-6}$

# Impact de la normalisation

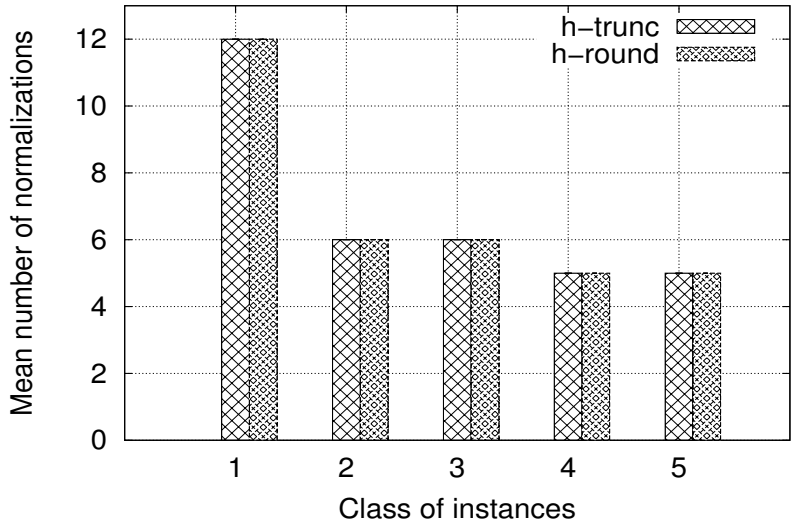



Le principe de normalisation commande de normaliser chaque fois que la précision initiale est dépassée. Quel est son impact dans le temps?

# Nombre de réajustements



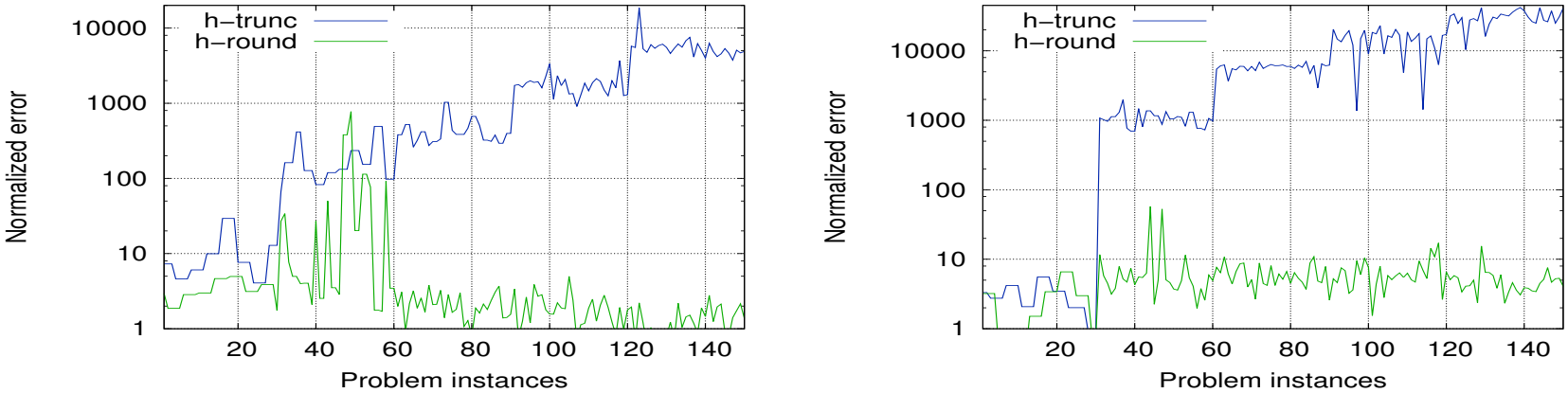

(a) Shipment, Initial precision = 6




(b) Disbursement, Initial precision = 3

Figure : Number of resizements per problem class

# Pertes d'information

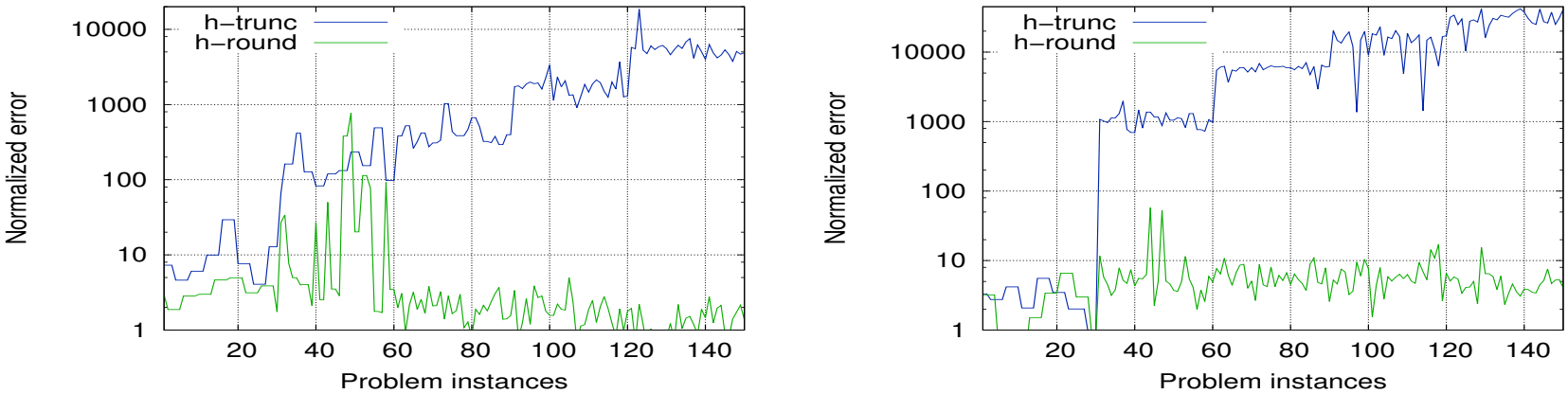



(a) Shipment, Initial precision = 6 (b) Disbursement, Initial precision = 3

Figure : Aggregated error per problem instance

$$norm(E) = \sum_{i=1}^{k} \sum_{z_u \in Cl_i} \left( \frac{|z_u - z'_i|}{|z'_i|} \right)^2$$

# Pertes d'information(II)




(a) Shipment, Initial precision = 6 (b) Disbursement, Initial precision = 3

Figure : Aggregated error per problem instance

Given three real numbers $x_1 < x_2 < x_3$, their median points will be $x' = x_2$ while the mean one will be $\bar{x} = \frac{x_1+x_2+x_3}{3}$ in the case where $x_1$ and $x_2$ are too small in comparison to $x_3$, the difference $\left(\frac{|x'-x_1|}{|x'|}\right)^2 + \left(\frac{|x'-x_3|}{|x'|}\right)^2$ can be arbitrarily large. In comparison, $\left(\frac{|\bar{x}-x_1|}{|\bar{x}|}\right)^2 + \left(\frac{|\bar{x}-x_3|}{|\bar{x}|}\right)^2 \leq \left(\frac{|3\bar{x}|}{|\bar{x}|}\right)^2 + \left(\frac{|3\bar{x}|}{|\bar{x}|}\right)^2 \leq 18$.

# Conclusion

- Réduction de graphes avec QoS probabilistes;
- Deux heuristiques *h-trunc* et *h-round*;
- Dilemme temps qualité avec *h-trunc* et *h-round*;
- Prise en compte d'autres modèles.