Acquisition de contraintes par requêtes de généralisation

Christian Bessiere[1], Remi Coletta[1], **Abderrazak Daoudi[1,2]**

Nadjib Lazaar[1], Younes Mechqrane[1] et El Houssine Bouyakhf[1]

1 CNRS, Université Montpellier 2, France

2 LIMIARF/FSR, Université Mohammed V-Agdal, Maroc.

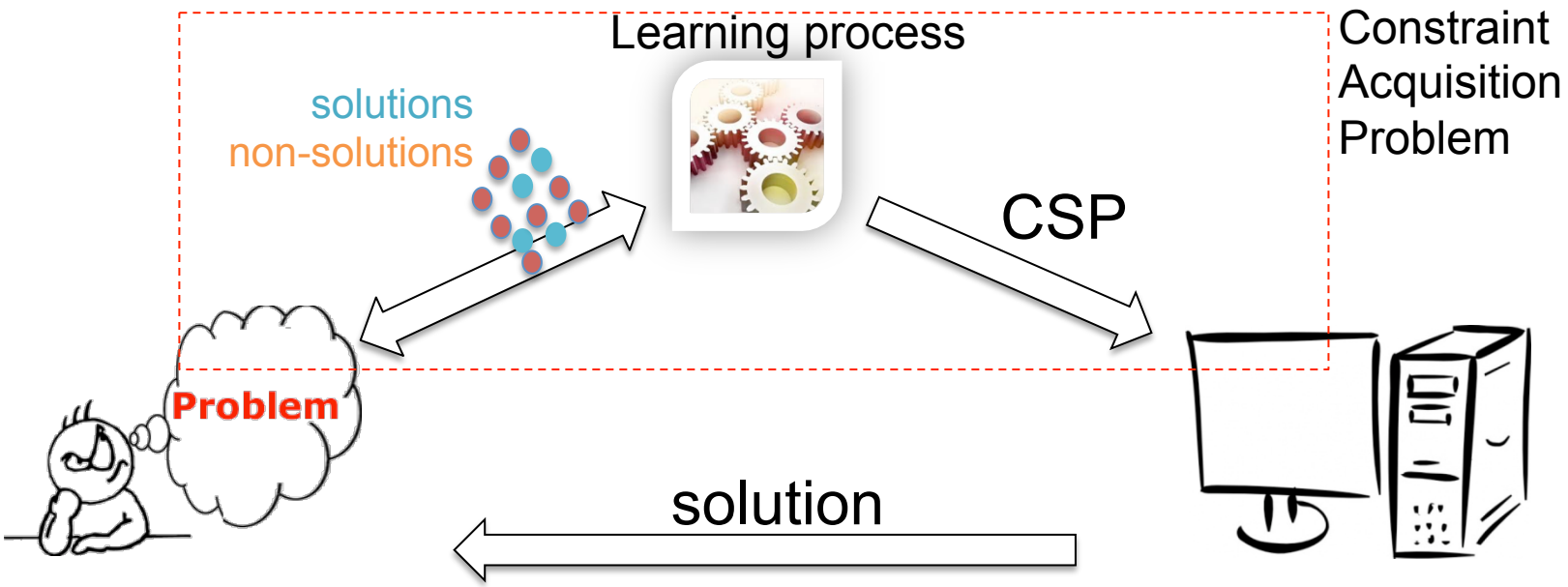
http://www.icon-fet.eu

JFPC'2014
Angers – 12 Juin 2014

Plan

- Acquisition de contraintes
- Généralisation de contraintes
  - Types de variables
  - Requête de généralisation
  - Algorithme GENACQ
  - Stratégies
  - Algorithme G-QUACQ
- Résultats
- Conclusion

Acquisition de contraintes



- Objectif : aider un novice à modéliser son problème

Acquisition de contraintes

- Entrée :
  - *(X,D)* : Vocabulaire
  - *B* : Biais (catalogue de contraintes)
  - C_T : Réseau cible
  - *E+, E-* : Ensembles d'exemples positifs et négatifs
- Sortie :
  - C_L *: Réseau appris*

$$(C_L \subseteq B) \wedge (C_L \equiv C_T)$$

Etat de l'art

- Conacq
  - Conacq.1 (apprentissage passif) [Bessiere et al. ECML05]
  - Conacq.2 (apprentissage actif) [Bessiere et al. IJCAI07]
  - Contraintes d'arité bornée dans un langage prédéfini
- ModelSeeker [Beldiceanu et Simonis, CP12]
  - Apprentissage passif
  - Basé sur le catalogue des contraintes globales (365 contraintes)

→ Le nombre de requêtes nécessaire pour converger vers le réseau de contraintes cible peut être exponentiel [Bessiere&Koriche2012]

Etat de l'art

- QUACQ [Bessiere et al. IJCAI13]
  - Apprentissage actif
  - Requêtes partielles à partir d'exemples négatifs
  - Le nombre de requêtes nécessaire pour apprendre une contrainte est logarithmique en la taille de l'exemple
  - Converge toujours vers le réseau cible en un nombre polynomial de requêtes

Faiblesses

- Si le problème a beaucoup de contraintes, un grand nombre de requêtes est nécessaire pour les apprendre toutes

  > 8000 requêtes pour apprendre le modèle du Sudoku

Généralisation de contraintes

- Requête de généralisation basée sur une agrégation des variables sous forme de types
- Algorithme de généralisation de contraintes qui peut être intégré dans n'importe quel système d'acquisition de contraintes
- Stratégies pour rendre notre approche plus efficace en terme de nombre de requêtes

Types de variables

- Dans les problèmes réels, les variables représentent souvent des composants qui peuvent être classées en différents types
- Problème d'emploi du temps scolaire :
  - les variables :
    - enseignants
    - des étudiants
    - des salles
    - …

→ Ces types sont souvent connus par l'utilisateur

Requête de généralisation

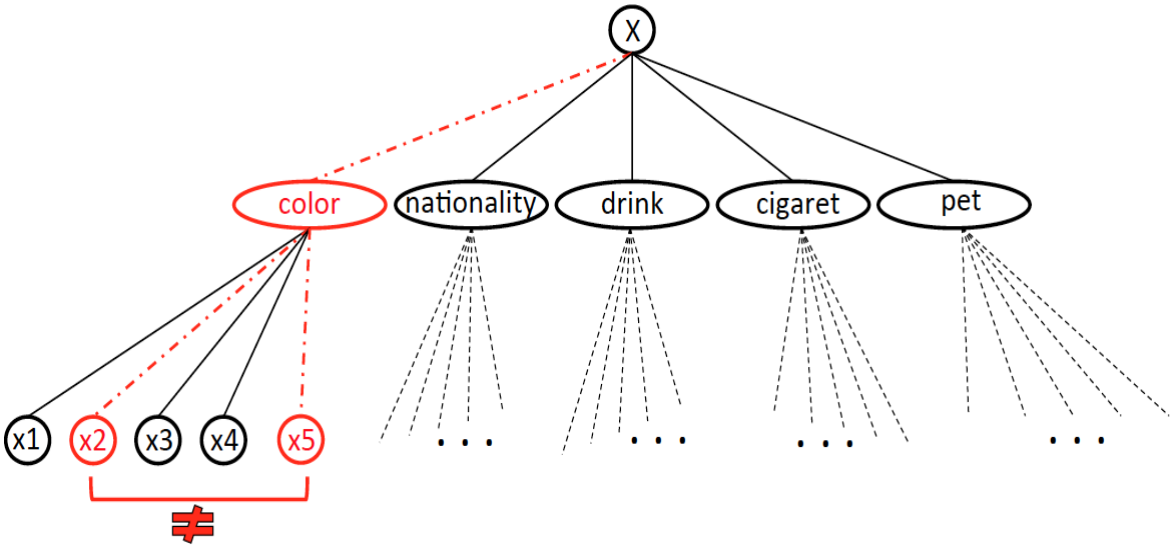
- Demande à l'utilisateur si une contrainte apprise peut être généralisée à d'autres portées de variables du même type que ceux de la contrainte apprise
  - Pr. X et Pr. Y ne peuvent pas être dans la même salle en même temps
  - Est-ce vrai pour toute paire d'enseignants ?

Algorithme GENACQ

- GENACQ demande à l'utilisateur de classifier les requêtes de généralisation à chaque fois qu'une contrainte est apprise

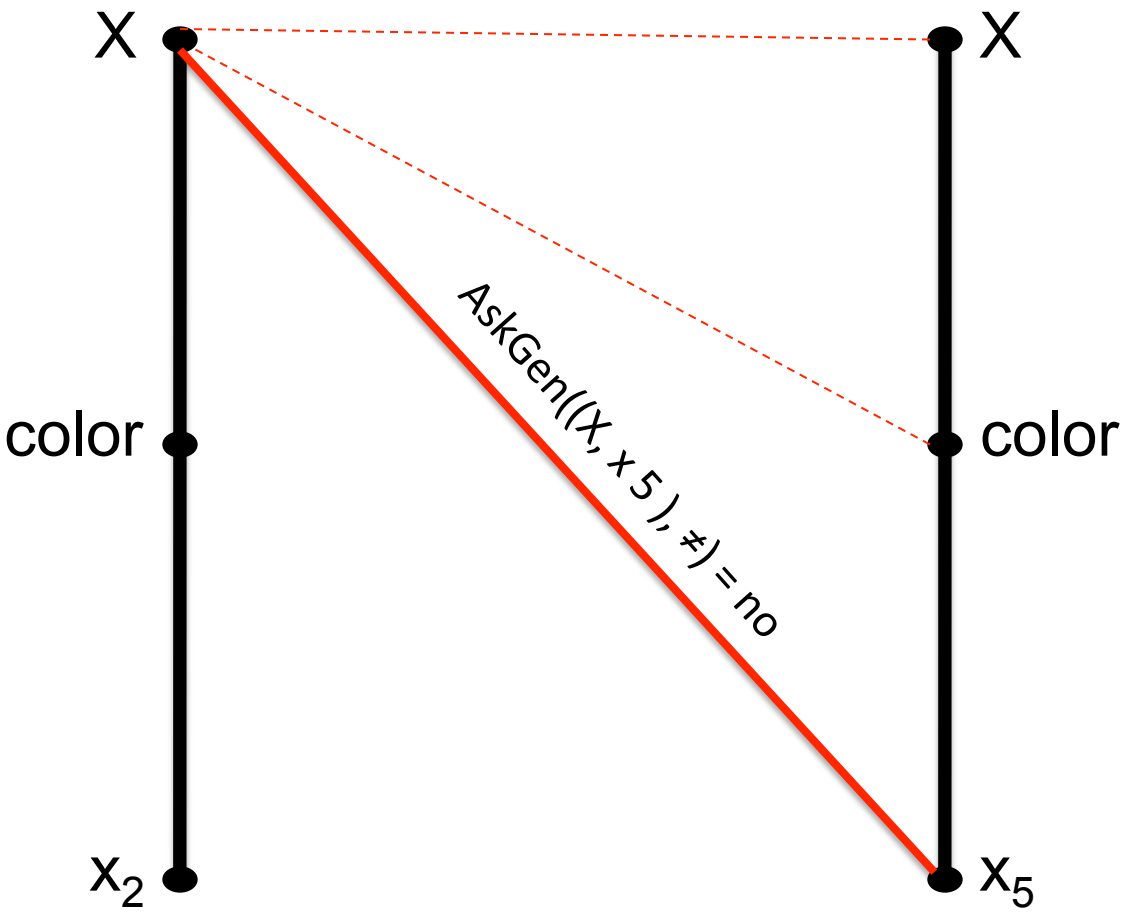
Exemple

- Problème de zèbre
- x2 ≠ x5 → contrainte apprise

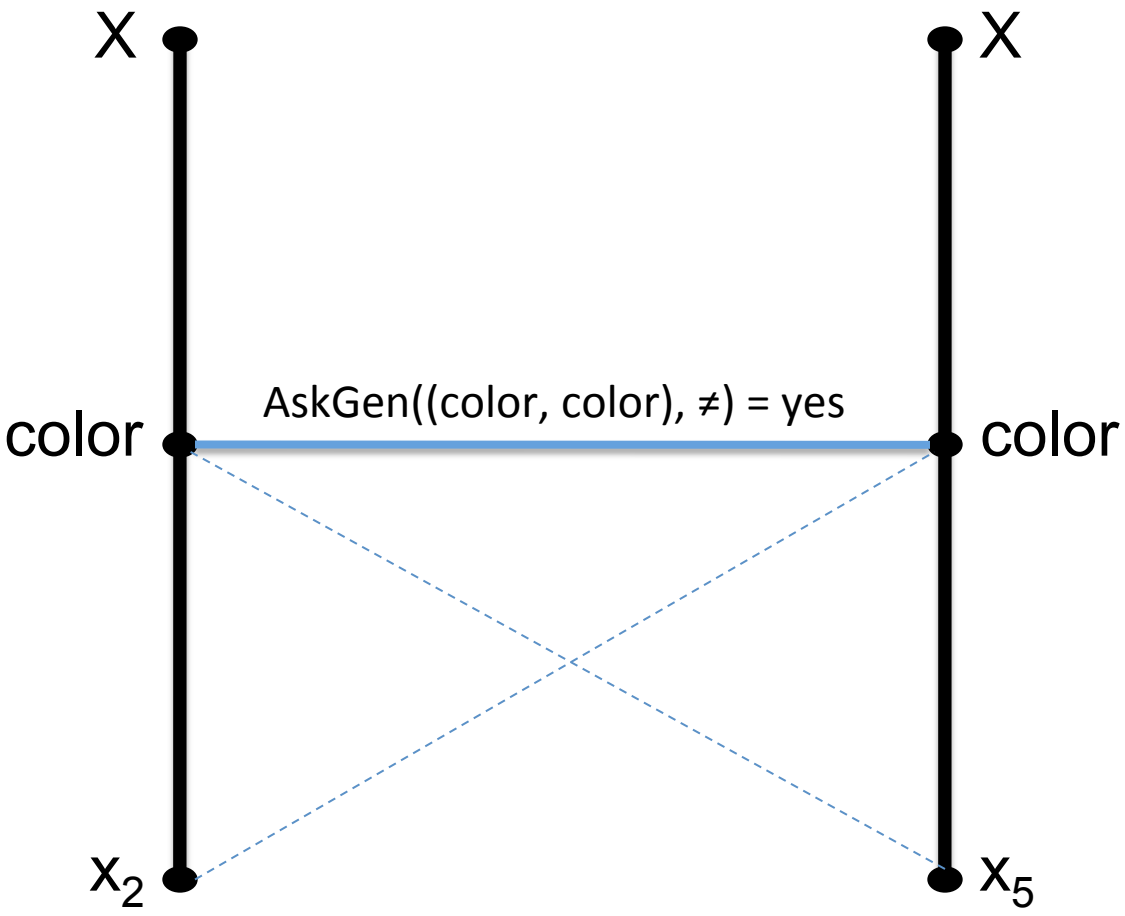


Variables et types pour le problème de zèbre

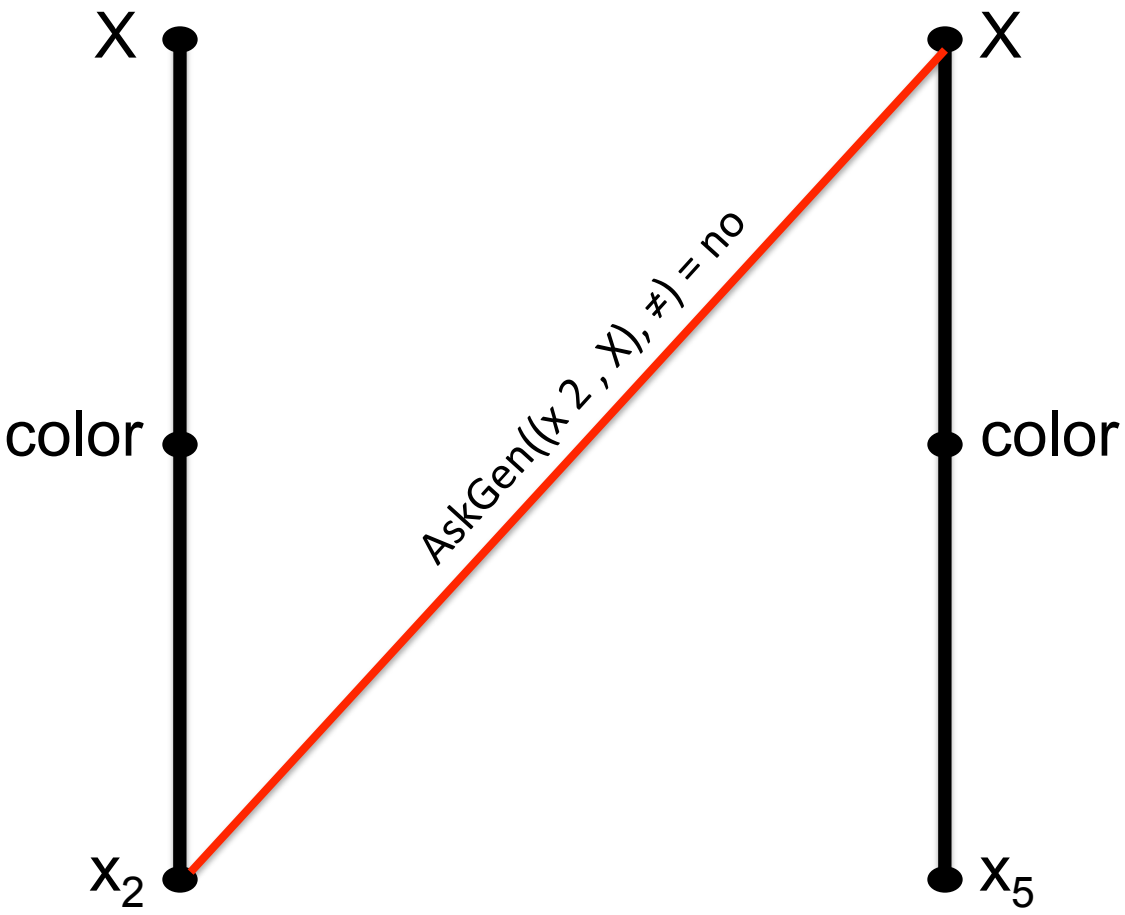
Exemple



Exemple



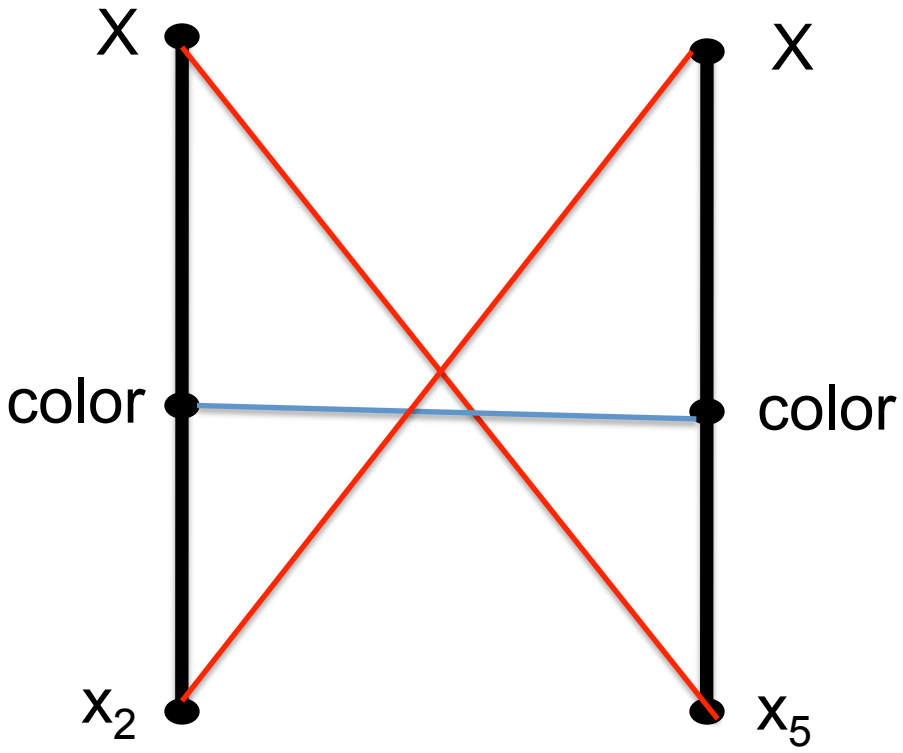
Exemple



→ x2 ≠ x5 peut être généralisée sur toutes les paires de variables couverte par la séquence (color, color)

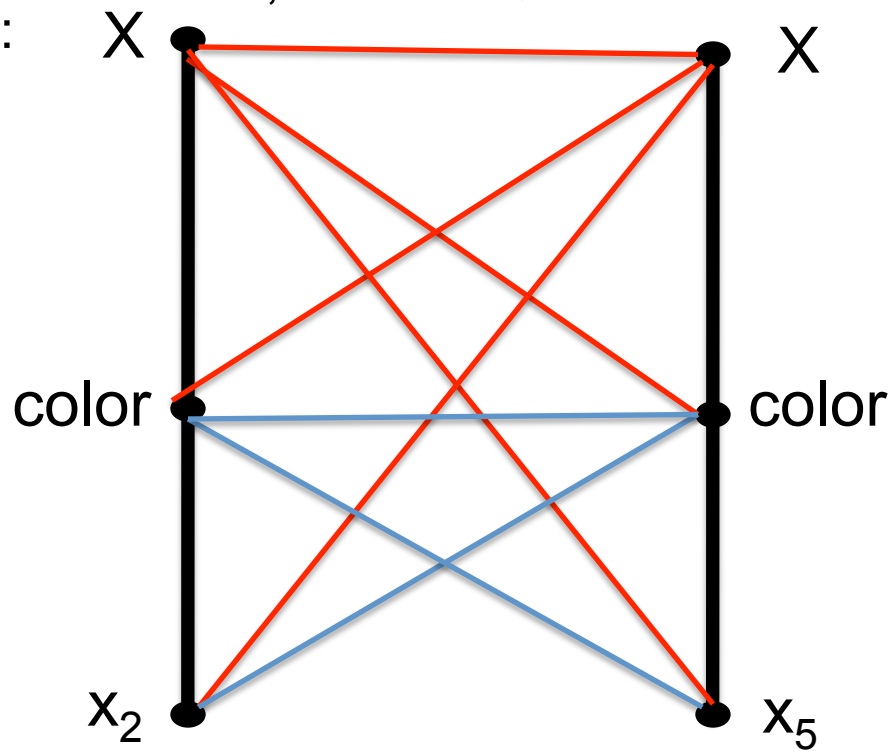
Stratégies

- Dans l'exemple précédent nous avons eu besoin de seulement 3 requêtes de généralisation pour converger sur la séquence maximale (color, color) sur laquelle ≠ s'applique



Stratégies

- En utilisant un autre ordre, GENACQ aura besoin de 8 requêtes de généralisation :



- Si l'on veut réduire le nombre de requêtes de généralisation, on peut se demander quelle stratégie utiliser ?

Stratégies

- max VAR → sélectionne une séquence s impliquant un nombre maximal de variables
- min VAR → sélectionne une séquence s impliquant un nombre minimal de variables
- max CST → sélectionne la séquence s maximisant le nombre de contraintes possibles (var, r) dans le biais
- min CST → sélectionne une séquence s minimisant le nombre de contraintes possibles (var, r) dans le biais
- Random → Il choisit au hasard une séquence

Stratégies

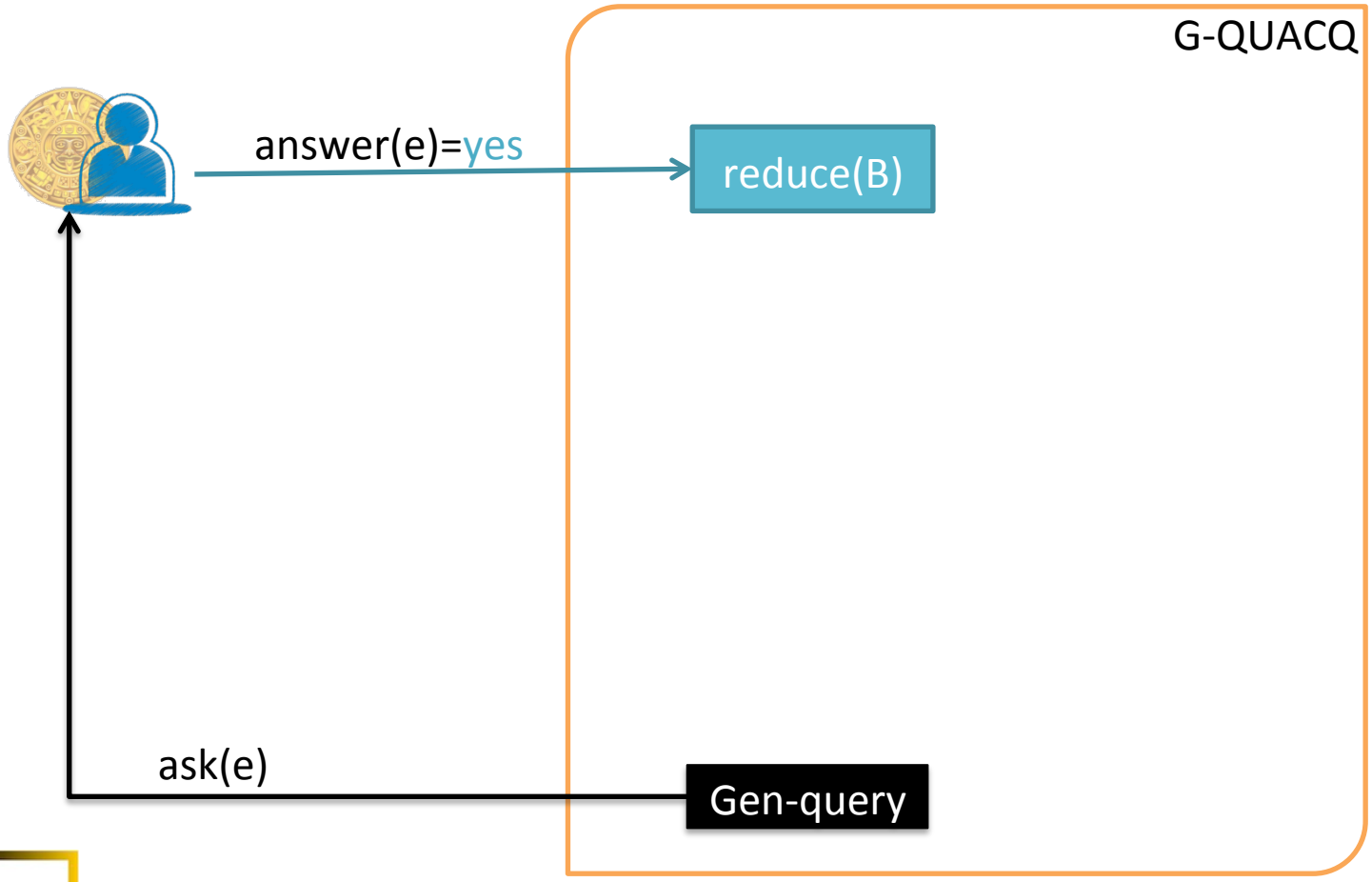
- Utilisation des Cutoffs → Nous mettons un seuil sur le nombre de réponses négatives consécutives afin d'éviter de vérifier des séquences peu prometteuses

G-QUACQ

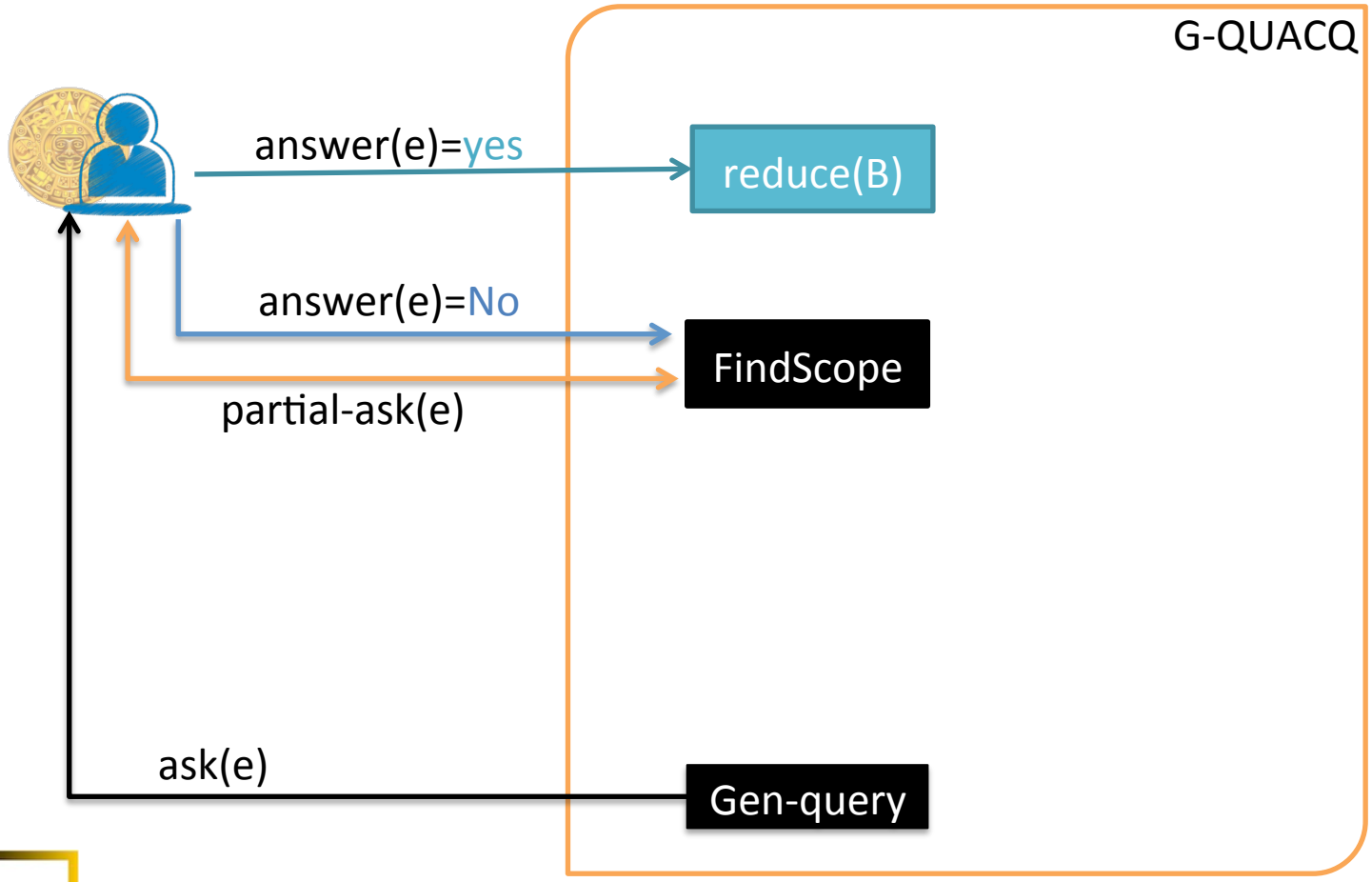
- GENACQ → une technique générique qui peut être intégrée dans n'importe quel système d'acquisition de contraintes

- G-QUACQ → un algorithme d'acquisition de contraintes obtenu en intégrant GENACQ dans QUACQ

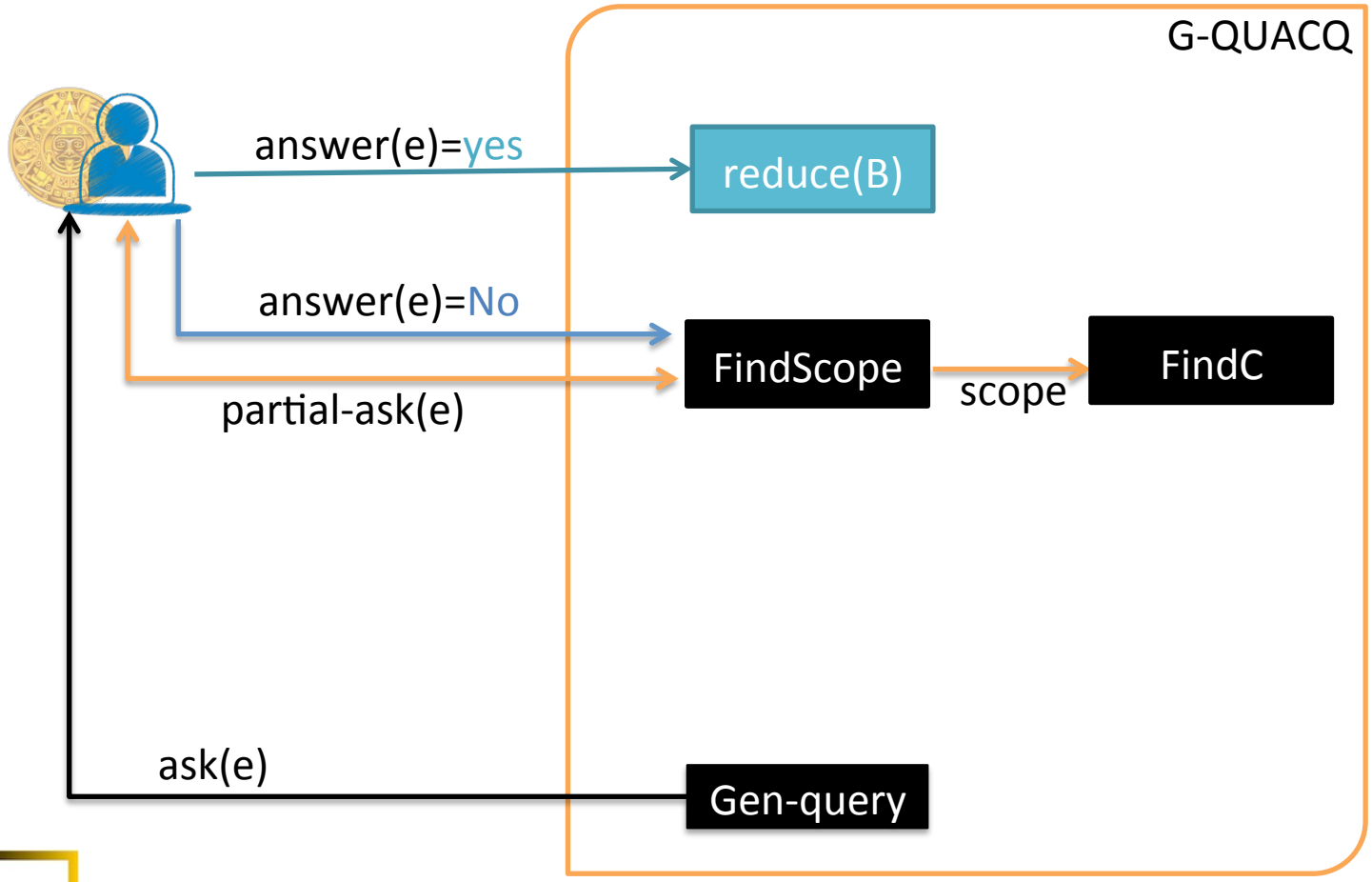
G-QUACQ



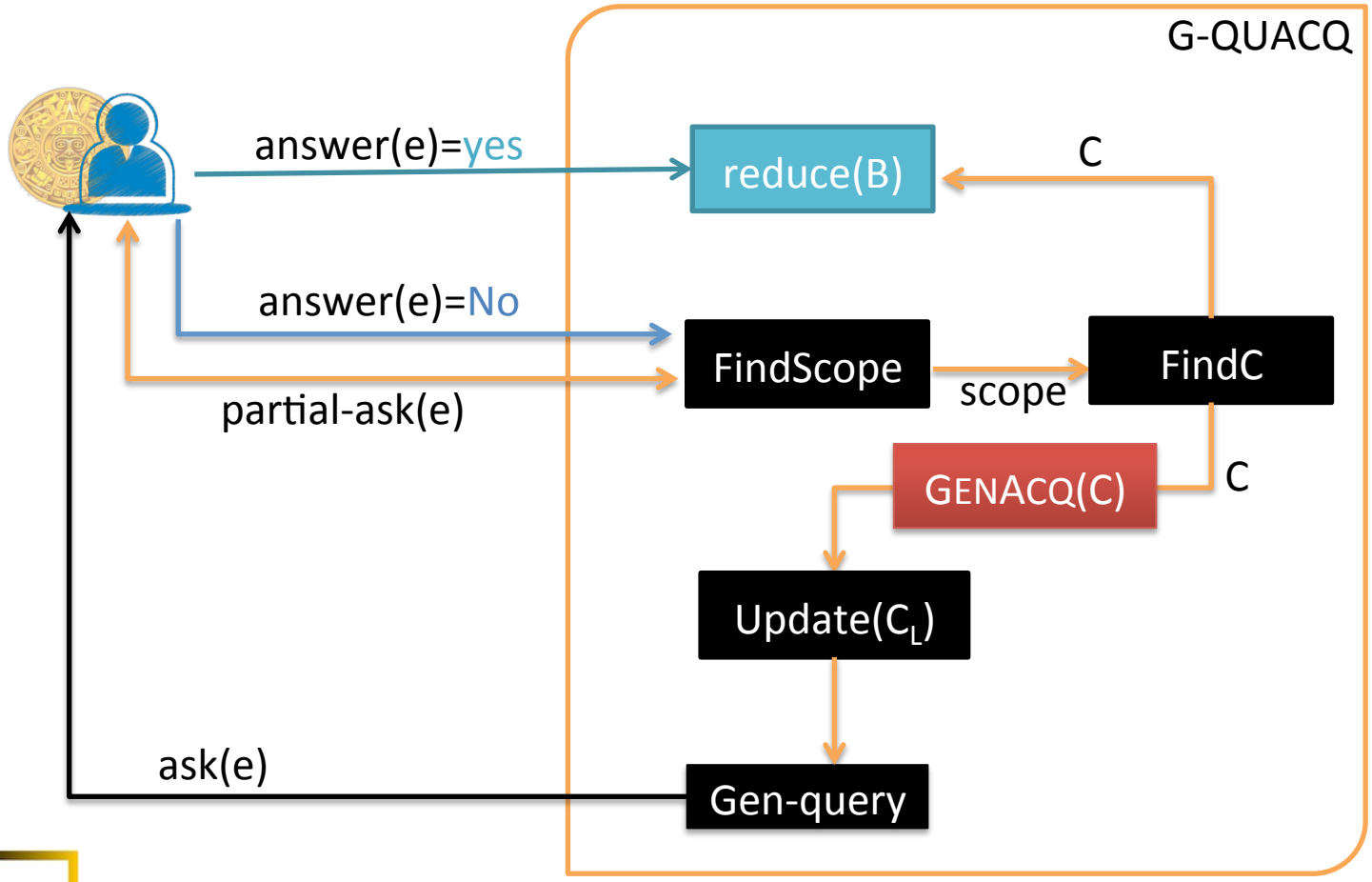
G-QUACQ


G-QUACQ
answer(e)=yes
reduce(B)
answer(e)=No
FindScope
partial-ask(e)
ask(e)
Gen-query

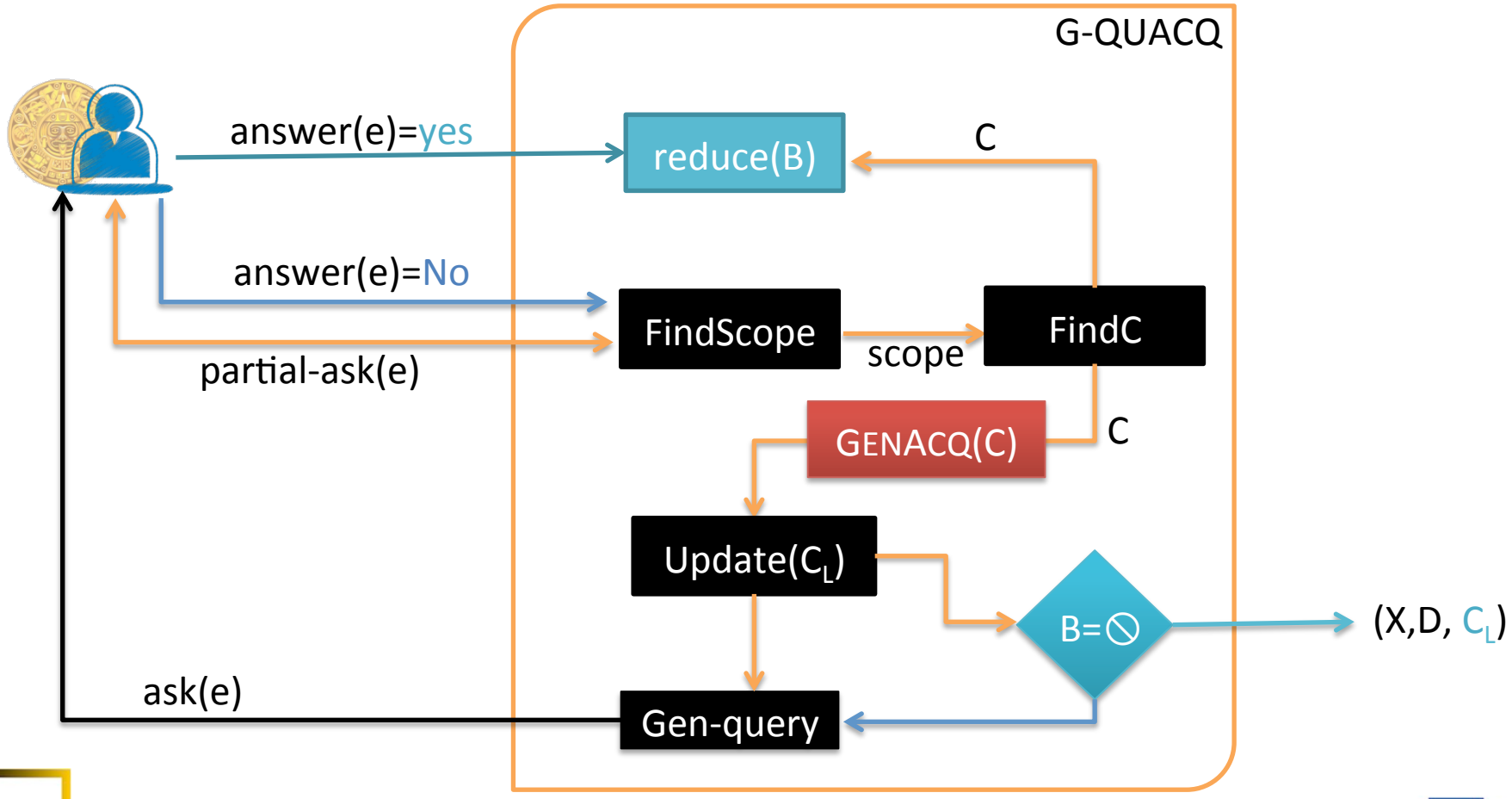
G-QUACQ


G-QUACQ
answer(e)=yes
reduce(B)
answer(e)=No
FindScope
scope
FindC
partial-ask(e)
ask(e)
Gen-query

G-QUACQ


G-QUACQ
answer(e)=yes
reduce(B)
C
answer(e)=No
FindScope
scope
FindC
partial-ask(e)
GENACQ(C)
C
Update(C_L)
ask(e)
Gen-query

G-QUACQ


G-QUACQ
answer(e)=yes
reduce(B)
C
answer(e)=No
FindScope
scope
FindC
partial-ask(e)
GENACQ(C)
C
Update(C_L)
B=∅
(X,D, C_L)
ask(e)
Gen-query

Résultats

QUACQ vs G-QUACQ

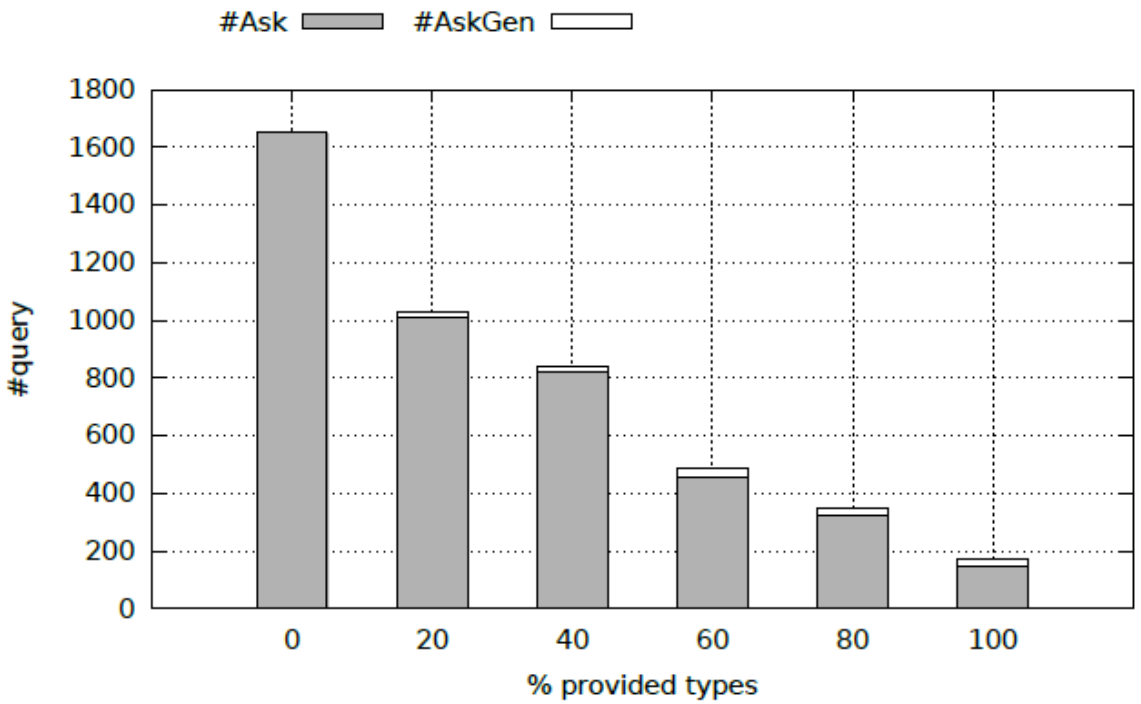
	QUACQ	G-QUACQ +random	
	#Ask	*#Ask*	*#AskGen*
Zebra	638	257	67
Sudoku	8645	260	166
Latin square	1129	117	60
RFLAP	1653	151	37
Purdey	173	82	31

Résultats

G-QUACQ avec les heuristiques et la stratégie cutoff sur le Sudoku

	cutoff	$\#Ask$	$\#AskGen$	$\#yes$	$\#no$
random	$+\infty$	260	166	42	124
min_VAR			90	21	69
min_CST			132	63	69
max_VAR			263	63	200
max_CST			247	21	226
min_VAR	3	260	75	21	54
	2		57	21	36
	1		**39**	21	18
min_CST	3	626	238	112	126
	2	679	231	132	99
	1	837	213	153	60

Résultats



G-QUACQ sur RLFAP lorsque le pourcentage de types fourni augmente.

Conclusion

- Nous avons proposé une nouvelle technique pour rendre l'acquisition de contraintes plus efficace en utilisant des informations sur les types des variables
- Nous avons introduit les requêtes de généralisation
- Algorithme GENACQ
- Nous avons proposé plusieurs heuristiques et stratégies pour sélectionner la requête de généralisation la plus prometteuse
- Algorithme G-QUACQ

Questions ?