Boosting Constraint Acquisition via Generalization Queries

Christian Bessiere¹
Nadjib Lazaar¹

Remi Coletta¹
Younes Mechqrane¹²

Abderrazak Daoudi¹²
El Houssine Bouyakhf²

¹University of Montpellier, France
²LIMIARF/FSR, University Mohammed V Agdal, Rabat, Morocco
Context
Question: How does the user write down the constraints of a problem?
Limitations: modelling constraint networks requires a fair expertise

Need: Simple way to build constraint models ➔ Modeller-assistant
• **Question:** How does the user write down the constraints of a problem?
• **Limitations:** modelling constraint networks requires a fair expertise

• **Need:** Simple way to build constraint models ➔ Modeller-assistant
• **How:** In a Machine Learning way (passive/active, offline/online, by reinforcement...)

---

**Context**

![Diagram](image)
• **Question:** How does the user write down the constraints of a problem?
• **Limitations:** modelling constraint networks requires a fair expertise

• **Need:** Simple way to build constraint models ➔ Modeller-assistant
• **How:** In a Machine Learning way (passive/active, offline/online, by reinforcement...)

---

**Constraint Acquisition Problem**

**Problem** ➔ **Learning process** ➔ **CSP** ➔ **solution**
Constraint Acquisition Problem

**Inputs:**
- \((X, D)\) : Vocabulary
- \(B\) : Basis (version space/possible constraints)
- \(C_T\) : Target Network
- \((E^+, E^-)\) : positives and negatives

**Output:**
- \(C_L\) : Learned network such that:
  \[ C_L \subset B : C_L \equiv C_T \]
Constraint Acquisition Systems

**CONACQ**

- Conacq1.0 (passive learning) [Bessiere et al. ECML05]
- Conacq2.0 (active learning) [Bessiere et al. IJCAI07]
Constraint Acquisition Systems

- **CONACQ**
  - Conacq1.0 (passive learning) [Bessiere et al. ECML05]
  - Conacq2.0 (active learning) [Bessiere et al. IJCAI07]

- **ModelSeeker** [Beldiceanu and Simonis, CP12]
  - A passive learning
  - Based on global constraint catalog (more than 400)
  - Bottom-up search
Constraint Acquisition Systems

- **CONACQ**
  - Conacq1.0 (passive learning) [Bessiere et al. ECML05]
  - Conacq2.0 (active learning) [Bessiere et al. IJCAI07]

- **ModelSeeker** [Beldiceanu and Simonis, CP12]
  - A passive learning
  - Based on global constraint catalog (more than 400)
  - Bottom-up search

- **QUACQ** [Bessiere et al. IJCAI13]
  - Active learning approach
  - Based on partial queries to elucidate the scope of the constraint to learn
Motivations

Limitation:
- Hard to put in practice:
  - ModelSeeker: cannot learn on unstructured problems
  - CONACQ and QUACQ: more than 8000 queries to learn the Sudoku model

Need:
- Reduce the dialogue with the user to make constraint acquisition more efficient in practice

How:
- Eliciting more information by asking complex queries to the user
A type is a subset of variables defined by the user as having a common property.

Example (School Timetabling Problem)

Can C1 be generalized to all Teachers, Rooms and Courses?
Let $c(x, y)$ a learned constraint and $X, Y$ are types of $x, y$:

**Generalization Query:** $AskGen((X, Y), c)$

The user says **yes** iff the constraint $c$ holds on all possible scope $(x_i, y_i) \in (X, Y)$

**Properties**

- Super-types: $X' \rightarrow X \rightarrow X''$
- Types: $Y' \rightarrow Y \rightarrow Y''$
- Sub-types: $X' \rightarrow Y'$
Let $c(x, y)$ a learned constraint and $X, Y$ are types of $x, y$:

- Generalization Query: $AskGen((X, Y), c)$

The user says yes iff the constraint $c$ holds on all possible scope

$$(x_i, y_i) \in (X, Y)$$

Properties

$$\text{AskGen}(X, Y, c) = \text{YES}$$

$$\text{AskGen}(X', Y', c) = \text{YES}$$

$$\text{AskGen}(X'', Y'', c) = \text{YES}$$
Let $c(x, y)$ a learned constraint and $X, Y$ are types of $x, y$:

- **Generalization Query**: $AskGen((X, Y), c)$

The user says **yes** iff the constraint $c$ holds on all possible scope

$$(x_i, y_i) \in (X, Y)$$

- **Properties**

  - $AskGen(\begin{array}{c} X' \\ \uparrow \\ X'' \end{array}, \begin{array}{c} Y' \\ \uparrow \\ Y'' \end{array}, c) = \text{NO}$

  - $AskGen(\begin{array}{c} X \\ \uparrow \\ X'' \end{array}, \begin{array}{c} Y \\ \uparrow \\ Y'' \end{array}, c) = \text{NO}$
**Inputs**
- A learned constraint
- Combination of possible types (i.e., table)

**Output**
- Set of constraints

![Diagram of Zebra Problem]

- Variables:
  - drink
  - cigaret
  - color
  - pet
  - nationality

- Types:
  - $X_1$
  - $X_2$
  - $X_3$
  - $X_4$
  - $X_5$
**Inputs**
- A learned constraint
- Combination of possible types (i.e., table)

**Output**
- Set of constraints

**Genacq**

**Inputs**
- Learned constraint: $x_2 \neq x_5$
- Table:

<table>
<thead>
<tr>
<th>#q = 4</th>
<th>$x_2$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>color</td>
<td>✔️</td>
</tr>
<tr>
<td>$x_2$</td>
<td>X</td>
<td>✗</td>
</tr>
<tr>
<td>color</td>
<td>$x_5$</td>
<td>✔️</td>
</tr>
<tr>
<td>color</td>
<td>color</td>
<td>✔️</td>
</tr>
<tr>
<td>color</td>
<td>X</td>
<td>✗</td>
</tr>
<tr>
<td>X</td>
<td>$x_5$</td>
<td>✗</td>
</tr>
<tr>
<td>X</td>
<td>color</td>
<td>✗</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>✗</td>
</tr>
</tbody>
</table>
**Inputs**
- A learned constraint
- Combination of possible types (i.e., table)

**Output**
- Set of constraints

**Zebra Problem**

**INPUTS**
- Learned constraint: $x_2 \neq x_5$
- Table:

<table>
<thead>
<tr>
<th></th>
<th>$x_2$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>$x_2$</td>
<td>$x_5$</td>
</tr>
<tr>
<td>color</td>
<td>$x_2$</td>
<td>$X$</td>
</tr>
<tr>
<td>color</td>
<td>$X$</td>
<td>$x_5$</td>
</tr>
<tr>
<td>color</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>color</td>
<td>$X$</td>
<td>$color$</td>
</tr>
</tbody>
</table>
Inputs
- A learned constraint
- Combination of possible types (i.e., table)

Output
- Set of constraints

Zebra Problem:

- Learned constraint: $x_2 \neq x_5$
- Table
- 9 constraints:

```
#q = 5
```

**INPUTS**

- drink
- cigaret
- color
- pet
- nationality

**OUTPUT**

- $x_1$
- $x_2$
- $x_3$
- $x_4$
- $x_5$
We implemented GENACQ and plugged it in the constraint acquisition system QUACQ for G-QUACQ version.

We Compared QUACQ to G-QUACQ on:
- Zebra problem (5 types of 5 variables)
- Sudoku (9 rows, 9 columns and 9 squares)
- Latin Square (5 rows and 5 columns)
- Radio link Frequency Assignment Problem (5 stations and 5 terminals)
- Purdey’s General Store Problem (4 families, 4 items, 4 payments)
<table>
<thead>
<tr>
<th></th>
<th>QUACQ</th>
<th>G-QUACQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Ask</td>
<td>#Ask</td>
</tr>
<tr>
<td>Zebra</td>
<td>638</td>
<td>257</td>
</tr>
<tr>
<td>Sudoku</td>
<td>8645</td>
<td>260</td>
</tr>
<tr>
<td>Latin square</td>
<td>1129</td>
<td>117</td>
</tr>
<tr>
<td>RFLAP</td>
<td>1653</td>
<td>151</td>
</tr>
<tr>
<td>Purdey</td>
<td>173</td>
<td>82</td>
</tr>
</tbody>
</table>

50% 95% 84% 88% 34%
Query Selection Heuristics

- Max constraints: 6 queries
- Max variables: 5 queries
- Min constraints: 5 queries
- Min variables: 5 queries
- Random

Cutoffs

- exit GENACQ before having proved the maximality
- Cutoff on the number of consecutive negative answers
**Result (1/3)**

G-QUACQ with heuristics and cutoff strategy on Sudoku

<table>
<thead>
<tr>
<th></th>
<th>cutoff</th>
<th>#Ask</th>
<th>#AskGen</th>
<th>#yes</th>
<th>#no</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>min_VAR</td>
<td>$+\infty$</td>
<td>260</td>
<td></td>
<td>90</td>
<td>21</td>
</tr>
<tr>
<td>min_CST</td>
<td></td>
<td></td>
<td></td>
<td>132</td>
<td>63</td>
</tr>
<tr>
<td>max_VAR</td>
<td></td>
<td></td>
<td></td>
<td>263</td>
<td>63</td>
</tr>
<tr>
<td>max_CST</td>
<td></td>
<td></td>
<td></td>
<td>247</td>
<td>21</td>
</tr>
</tbody>
</table>

|        | 3     | 260  | 75      | 21   | 54  |
| min_VAR| 2     |      | 57      | 21   | 36  |
|        | 1     |      | 39      | 21   | 18  |
| min_VAR| 3     | 626  | 238     | 112  | 126 |
| min_CST| 2     | 679  | 231     | 132  | 99  |
|        | 1     | 837  | 213     | 153  | 60  |
G-QUACQ with random, min_VAR, and cutoff=1 on Zebra, Latin square, RLFAP, and Purdey

<table>
<thead>
<tr>
<th></th>
<th>#Ask</th>
<th>#AskGen</th>
<th>#yes</th>
<th>#no</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Zebra</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>257</td>
<td></td>
<td>67</td>
<td>10</td>
</tr>
<tr>
<td>min_VAR</td>
<td></td>
<td>48</td>
<td>5</td>
<td>43</td>
</tr>
<tr>
<td>min_VAR+cutoff=1</td>
<td></td>
<td>23</td>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Latin square</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>117</td>
<td></td>
<td>60</td>
<td>16</td>
</tr>
<tr>
<td>min_VAR</td>
<td></td>
<td>34</td>
<td>10</td>
<td>24</td>
</tr>
<tr>
<td>min_VAR+cutoff=1</td>
<td></td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
G-QUACQ with random, min_VAR, and cutoff=1 on Zebra, Latin square, RLFAP, and Purdey

<table>
<thead>
<tr>
<th></th>
<th>#Ask</th>
<th>#AskGen</th>
<th>#yes</th>
<th>#no</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RLFAP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>151</td>
<td>37</td>
<td>16</td>
<td>21</td>
</tr>
<tr>
<td>min_VAR</td>
<td></td>
<td>41</td>
<td>14</td>
<td>27</td>
</tr>
<tr>
<td>min_VAR +cutoff=1</td>
<td></td>
<td>22</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td><strong>Purdey</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>82</td>
<td>31</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>min_VAR</td>
<td></td>
<td>24</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>min_VAR +cutoff=1</td>
<td></td>
<td>12</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>
Conclusions

- Generalization query based on types of variables
- GENACQ algorithm
- Several heuristics and strategies to select the good candidate generalization query
- Can be plugged in any active constraint acquisition system
- Results by plugging GENACQ in the QUACQ acquisition System