This interactive constraint acquisition system was developed as part of the PhD of Dimosthenis C. Tsouros. So far it has been used for the evaluation of the various constraint acquisition algorithms it supports, but it is under further development in order to incorporate a wider range of constraints and become a system that can be utilised by the CP community. The system that was used for the experiments reported in the PhD thesis has been developed in C++ by Dimosthenis C. Tsouros. This system exploits a custom written CP solver. However, there is also a working implementation in JAVA, exploiting the Choco solver. This second system is also under further development.

The executable file of the C++ implementation can be found in the following link: https://github.com/Dimosts/Constraint_Acquisition. It was compiled in UBUNTU 18.04. The code is not yet available publicly, but on request.

The system supports the following Constraint Acquisition algorithms:

- Quacq [1]
- MultiAcq [9]
- MQuAcq [4-5]
- MQuAcq-2 [6]
- MQuAcq-2-OM1 [7]
- MQuAcq-2-OM2 [7]
- PrefAcq [8]

It also supports all the functions/subsystems that were presented in [1-9].

At the end of any algorithm's execution the system exports the learned network and writes in a file some important metrics for the evaluation of the algorithm. The evaluation metrics are written in the file named "results_<algorithm name>_<benchmark name>".

Parameters

<algorithm>	Supports as options the following algorithms - Quacq [1] - MultiAcq [9] - MQuAcq [4-5] - MQuAcq-2 [6] - PrefAcq [8]
-u	If the oracle answering the queries is a human user (default)
-nu	if the oracle is a software system.
-h <#>	determines the variable ordering heuristic. Available options: - 0 Use of lexicographic variable ordering - 1 Use of *dom* variable ordering heuristic - 2 Use of *dom/wdeg* variable ordering heuristic - 3 Use of *bdeg* variable ordering heuristic [5]
-domh <#>	determines the value ordering heuristic. Available options: - 0 Use of lexico heuristic, i.e. choose the values in a lexicographic order

	- 1 Use of random heuristic, i.e. choose the values in a random order - 2 Use of max_v heuristic, i.e. choose the value violating a maximum number of constraints from the bias [5].
-f <benchmark>	Choice of the benchmark to use. There must be a folder with the same name in the same path with the exe file. The folder must contain a <benchmark>_var file defining the variables existing in the problem, a <benchmark>_dom file defining the domains, a <benchmark>_con file if -nu is used, in order to let the system answer the queries without the need of a human user, a <benchmark>_scon if prefacq is used with -nu in order to let the system answer the queries without the need of a human user, a <benchmark>_cl file defining the initial constraint network we want to give to the algorithm if -I is used.
-maxb	use of maxb heuristic for the generation of the queries i.e. return the (partial solution with the maximum constraints of the bias violated (at least 1) (cutoff 1 sec) [5]
-solp	use of solp heuristic for the generation of the queries i.e. return the (partial) solution with the maximum number of variables instantiated (cutoff 1 sec) [3]
-sol	use of sol heuristic for the generation of the queries, i.e. return first solution found [1]
-min	use of min heuristic for the generation of the queries i.e. return the solution with the minimum constraints of the bias violated (at least 1) (cutoff 1 sec)
-max	use of max heuristic for the generation of the queries. i.e. return the solution with the maximum constraints of the bias violated (at least 1) (cutoff 1 sec) [1]
-I, --initial-cl	giving the system a initial constraint network of the problem to complete (we must have a <benchmark>_cl file in the benchmark folder)
-fs	Choose which FindScope function to use - 1 FindScope function from [1] - 2 FindScope function from [4-5] - 3 FindScope function from [3]
-fc	Choose which FindC function to use - 0 FindC function from [1] - 1 FindC function from [2] - 2 Slightly improved FindC function from [2] - 3 FindC function from [3]
-om, -lmq, --omissions	Use of omissions handling (implemented only for MQuAcq-2 for now) - 1 use of MQuAcq-OM1 [7] - 2 use of MQuAcq-OM2 [7]

[1] Christian Bessiere, Remi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, Toby Walsh, et al. Constraint acquisition via partial queries. In IJCAI, volume 13, pages 475–481, 2013.

[2] Christian Bessiere, Abderrazak Daoudi, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Younes Mechqrane, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. New approaches to constraint acquisition. In Data mining and constraint programming, pages 51–76. Springer, 2016

[3] Christian Bessiere, Clement Carbonnel, Anton Dries, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, Kostas Stergiou, Dimosthenis C. Tsouros, Toby Walsh, "Partial Queries for Constraint Acquisition." *arXiv preprint arXiv:2003.06649* (2020).

[4] Dimosthenis C. Tsouros, Kostas Stergiou, and Panagiotis G. Sarigiannidis. Efficient methods for constraint acquisition. In 24th International Conference on Principles and Practice of Constraint Programming, 2018

[5] Dimosthenis C Tsouros and Kostas Stergiou. Efficient multiple constraint acquisition. Constraints, 25(3):180–225, 2020.

[6] Dimosthenis C Tsouros, Kostas Stergiou, and Christian Bessiere. Structuredriven multiple constraint acquisition. In International Conference on Principles and Practice of Constraint Programming, pages 709–725. Springer, 2019.

[7] Dimosthenis C Tsouros, Kostas Stergiou, and Christian Bessiere. Omissions in constraint acquisition. In International Conference on Principles and Practice of Constraint Programming, pages 935–951. Springer, 2020

[8] Dimosthenis C. Tsouros and Kostas Stergiou. Learning max-csps via active constraint acqusition (to appear). In 27th International Conference on Principles and Practice of Constraint Programming, 2021.

[9] Robin Arcangioli, Christian Bessiere, and Nadjib Lazaar. Multiple constraint aquisition. In IJCAI: International Joint Conference on Artificial Intelligence, pages 698–704, 2016.