Constraint Acquisition with Recommendation Queries

Abderrazak Daoudi
U. Mohammed V
Rabat, Morocco

Younes Mechqrane
U. Mohammed V
Rabat, Morocco

Christian Bessiere
U. of Montpellier
France

Nadjib Lazaar
U. of Montpellier
France

El Houssine Bouyakhf
U. Mohammed V
Rabat, Morocco

Abstract

Constraint acquisition systems assist the non-expert user in modeling her problem as a constraint network. Most existing constraint acquisition systems interact with the user by asking her to classify an example as positive or negative. Such queries do not use the structure of the problem and can thus lead the user to answer a large number of queries. In this paper, we propose PREDICT&ASK, an algorithm based on the prediction of missing constraints in the partial network learned so far. Such missing constraints are directly asked to the user through *recommendation* queries, a new, more informative kind of queries. PREDICT&ASK can be plugged in any constraint acquisition system. We experimentally compare the QUACQ system to an extended version boosted by the use of our recommendation queries. The results show that the extended version improves the basic QUACQ.

1 Introduction

Constraint programming (CP) allows effective solving of combinatorial problems in many areas, such as planning and scheduling. However, modeling a combinatorial problem using constraints is a fastidious task that requires significant expertise in CP [Freuder, 1999].

To make constraint programming accessible to novices, several constraint acquisition systems have been introduced in the last decade [Bessiere *et al.*, 2005; 2007; Beldiceanu and Simonis, 2012; Lallouet *et al.*, 2010; Shchekotykhin and Friedrich, 2009]. These systems either need an expert to validate the learned model or need an exponential number of queries to converge on the target constraint network [Bessiere *et al.*, 2015]. Recently, a system polynomial in terms of queries, called QUACQ, has been proposed [Bessiere *et al.*, 2013]. QUACQ iteratively generates *partial* queries (that is, partial assignments of the variables) and asks the user to classify them. When the answer of the user is *yes*, QUACQ reduces the search space by removing constraints that reject the positive example. In the case of a negative answer, QUACQ focuses on a constraint in a number of queries logarithmic in the size of the example. This key component allows QUACQ to converge on the target network in a polynomial number of queries. Despite this good theoretical bound, QUACQ may require a lot of queries to learn the target constraint network, especially when the problem is highly structured and involves a large number of constraints. For instance, the user has to classify more than 8000 queries to get the Sudoku model.

The next challenge to constraint acquisition is to reduce the dialog length between the user and the learner or, in other words, to reduce the number of asked queries to get the target model. This paper presents a generic approach to constraint acquisition which is centered on the following question: Given the constraint graph learned so far, can we infer which new constraints are more likely to belong to the target constraint network? We formalize this question as a link prediction problem in the partial constraint graph learned so far. Furthermore, we introduce a new concept of query, called *recommendation* query. Borrowing techniques from the link prediction field, a recommendation query asks the user whether or not a predicted constraint belongs to the target constraint network. To deal with recommendation queries, we propose a new constraint recommender algorithm called PREDICT&ASK, which we plugged into the QUACQ constraint acquisition system leading to the P-QUACQ algorithm. We experimentally evaluated the benefit of our approach on several benchmark problems. The results show that P-QUACQ significantly improves the basic QUACQ algorithm in terms of number of queries.

The rest of the paper is organized as follows. Section 2 gives the necessary material to understand the technical presentation. Section 3 describes the constraint recommender algorithm. We illustrate the idea behind our constraint recommender algorithm through an example in Section 4. In Section 5, several predictor techniques are presented. Section 6 presents the experimental results we obtained when comparing P-QUACQ to the basic QUACQ. Section 7 presents the related work. Section 8 concludes the paper.

2 Background

The common knowledge shared between the user and the learner is the *vocabulary*. This vocabulary is represented by a (finite) set of variables X and domains $D = \{D(x_1), \ldots, D(x_n)\}$ over $\mathbb{Z}$. A constraint c is defined by a pair $(var(c), rel(c))$, where $rel(c)$ is the relation specifying which sequences of $|var(c)|$ values are allowed for the variables $var(c)$. $var(c)$ is called the scope of c and $|var(c)|$ the arity of $rel(c)$. Without loss of generality, we restrict ourselves to binary constraints. Combinatorial problems are represented with *constraint networks*. A constraint network is a set C of constraints on the vocabulary (X, D). An example e is a (partial/complete) assignment on a set of variables $var(e) \subseteq X$. e is rejected by a constraint c (i.e., $e \not\models c$) iff $var(c) \subseteq var(e)$ and the projection $e[var(c)]$ of e on $var(c)$ is not in c. A complete assignment e of X is a solution of C iff for all $c \in C$, c does not reject e. We denote by $sol(C)$ the set of solutions of C.

In addition to the vocabulary, the learner owns a *language* Γ of relations from which it can build constraints on specified sets of variables. A *constraint basis* is a set B of constraints built from the constraint language Γ on the vocabulary (X, D). Formally speaking, $B = \{c \mid (var(c) \subseteq X) \wedge (rel(c) \in \Gamma)\}$.

In terms of machine learning, a *concept* is a Boolean function over $D^X = \Pi_{x_i \in X} D(x_i)$, that is, a map that assigns to each example $e \in D^X$ a value in $\{0, 1\}$. We call *target concept* the concept f_T that returns 1 for e if and only if e is a solution of the problem the user has in mind. In a constraint programming context, the target concept is represented by a *target network* denoted by C_T. A *query* $Ask(e)$, with $var(e) \subseteq X$, is a classification question asked to the user, where e is an assignment in $D^{var(e)} = \Pi_{x_i \in var(e)} D(x_i)$. A set of constraints C *accepts* an assignment e if and only if there does not exist any constraint $c \in C$ rejecting e. The answer to $Ask(e)$ is *yes* if and only if C_T accepts e.

In this paper we introduce a new kind of query, *recommendation queries* $AskRec(c)$, which ask the user whether or not the constraint c belongs to the target constraint network C_T. It is answered yes if and only if c belongs to C_T.

3 PREDICT&ASK Algorithm

In this section, we present our constraint recommender PREDICT&ASK algorithm. The idea behind this algorithm is to predict missing constraints in the partial network learned so far, and then to recommend the predicted constraints to the user through recommendation queries.

3.1 Description of PREDICT&ASK

The algorithm PREDICT&ASK takes as argument the set of constraints C learned so far, a relation r, and the predictor $score$ that corresponds to the strategy used to assign a cost to a candidate constraint for recommendation. The algorithm uses the local data structure Δ which contains all constraints that are candidate for recommendation.

PREDICT&ASK starts by initializing L to the empty set (line 1). The set L will contain the output of PREDICT&ASK, that is all constraints learned by prediction plus recommendation query. In line 4, we build the constraint graph $G = (Y, E)$ restricted to the relation r. The counter $\#No$ counts the number of consecutive times recommendation queries have been classified negative by the user. It is initialized to zero at line 5. We put in Δ all constraints that are candidate for recommendation.

Algorithm 1: PREDICT&ASK

Input: C: a set of constraints,
r: a relation,
$score \in \{AA, LHN\}$: a prediction strategy
Output: L: a set of predicted constraints

```
1  L ← ∅;
2  Y ← ⋃ var(c) s.t. (c ∈ C ∧ rel(c) = r)
3  E ← {(x, y) | c ∈ C ∧ rel(c) = r ∧ var(c) = (x, y)}
4  G ← (Y, E)
5  #No ← 0
6  Δ ← {((x, y), r) ∈ B | (x, y) ∈ Y² \ E}
7  while Δ ≠ ∅ ∧ #No < α do
8      pick ((x, y), r) in Δ that maximizes score((x, y), G)
9      if AskRec((x, y), r) = yes then
10         L ← L ∪ {((x, y), r)}
11         E ← E ∪ (x, y)
12         #No ← 0
13     else
14         B ← B \ {((x, y), r') | r ⊆ r'}
15         #No + +
16 return L;
```

In the main loop of PREDICT&ASK (line 7), for each iteration, we pick a constraint from Δ such that its score is maximum (line 8). A constraint with a high score means that it is likely that this constraint belongs to the target constraint network. Hence, PREDICT&ASK asks a recommendation query on $((x, y), r)$ (line 9). If the user says 'yes', $((x, y), r)$ is a constraint of the target network. Hence, we put $((x, y), r)$ in L (line 10). We also add the edge (x, y) to E to be taken into account in the next iteration when computing the $score$. In line 12, we reinitialize $\#No$ to zero. If the user says 'no', we remove from B the constraint $((x, y), r)$ (line 14) and we increment $\#No$ (line 15). The loop ends when Δ is empty or when $\#No$ reaches the given threshold α, and we return L (line 16).

3.2 Using Recommendation in QUACQ

PREDICT&ASK is a generic constraint recommender algorithm that can be plugged into any constraint acquisition system. In this section, we present P-QUACQ (Algorithm 2) where we incorporate PREDICT&ASK into the QUACQ system.

P-QUACQ initializes the constraint network C_L to the empty set (line 1). When C_L is unsatisfiable (line 3), the space of possible networks collapses because there does not exist any subset of the given basis B that is able to correctly classify the examples already asked to the user. In line 4, P-QUACQ computes a complete assignment e satisfying C_L and

Algorithm 2: P-QUACQ = QUACQ + PREDICT&ASK

Input: $score \in \{AA, LHN\}$: a predictor strategy
Output: C_L: a set of learned constraints

```
1  C_L ← ∅ ;
2  while true do
3      if sol(C_L) = ∅ then return "collapse";
4      choose e in D^X accepted by C_L and rejected by B
5      if e = nil then return "convergence on C_L";
6      if Ask(e) = yes then
7          B ← B \ κ_B(e);
8      else
9          c ← FindC(e, FindScope(e, ∅, X, false));
10         if c = nil then return "collapse";
11         else
12             C_L ← C_L ∪ {c};
13             C_L ← C_L ∪ PREDICT&ASK(C_L, rel(c), score);
```

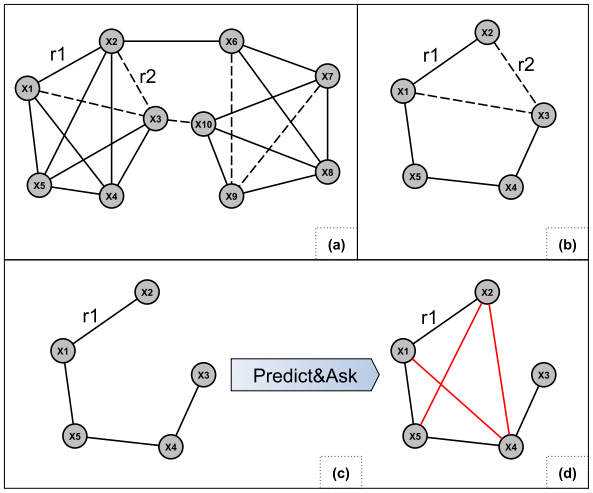

Figure 1: PREDICT&ASK on the illustrative example.

violating at least one constraint from B. If such an example does not exist (line 5), then all constraints in B are implied by C_L, and the algorithm has converged. Otherwise, we propose the example e to the user, who will answer by *yes* or *no* (line 6). If the answer is *yes*, we can remove from B the set $\kappa_B(e)$ of all constraints in B that reject e (line 7). If the answer is *no*, we are sure that e violates at least one constraint of the target network C_T. We then call the function `FindScope` to discover the scope of one of these violated constraints. Here, `FindScope` acts in a dichotomous manner and asks a number of queries logarithmic in the size of the example. `FindC` selects which constraint with the given scope is violated by e (line 9). If no constraint is returned (line 10), this is a condition for collapsing as we could not find in B a constraint rejecting one of the negative examples. Otherwise, we know that the constraint c returned by `FindC` belongs to the target network C_T, we then add it to the learned network C_L (line 12). Note that `FindScope` and `FindC` functions are used exactly as they appear in [Bessiere *et al.*, 2013]. Afterwards, we call PREDICT&ASK to mine the learned constraint network C_L in order to predict and recommend missing constraints that may belong to the target network. P-QUACQ updates C_L by adding all learned constraints (line 13).

3.3 Complexity Analysis

Let us now give the theoretical upper bound of the new constraint acquisition system P-QUACQ.

Theorem 1. *Given a constraint basis B built from a language Γ of bounded arity, and a target network C_T,* P-QUACQ *uses $O(C_T.(log|X| + \Gamma) + |B|)$ queries to prove convergence or to collapse.*

Proof. By construction, P-QUACQ inherits the correctness of QUACQ, and thus, it always finishes by proving convergence or collapsing. As for its complexity, P-QUACQ asks partial queries (line 6 of P-QUACQ) and recommendation queries (line 9 of PREDICT&ASK). By construction, the number of partial queries in P-QUACQ is bounded above by the number of partial queries of pure QUACQ, that is, $O(C_T.(log|X| + \Gamma) + |B|)$ [Bessiere *et al.*, 2013]. Concerning recommendation queries, we know that they are asked on constraints that are in B and not in C_L (Algorithm 1, lines 6 and 8). Furthermore, a recommendation query cannot be asked twice on the same constraint as, whatever the answer, the constraint is put in C_L (*yes* answer, Algorithm 1, line 10 and Algorithm 2, line 13) or removed from B (*no* answer, Algorithm 1, line 14). As a result the number of recommendation queries asked by PREDICT&ASK is in $O(|B|)$ and the number of queries asked by P-QUACQ is in $O(C_T.(log|X| + \Gamma) + |B|)$. □

4 An Illustrative Example

In this section, we illustrate our constraint recommender algorithm PREDICT&ASK through an example. Figure 1(a) shows the constraint network of the problem that the user has in mind. This problem involves 10 variables and 21 binary constraints. Two relations are used, noted r_1 and r_2 in Figure 1. Figure 1(b) shows the constraint network partially learned by QUACQ. Suppose that the last constraint learned using QUACQ was $((x_1, x_2), r_1)$. At that point, we want to recommend potential constraints on which the relation r_1 may hold. PREDICT&ASK builds a partial network limited to the relation r_1 (Figure 1(c)), and then computes the set Δ of all candidate constraints that may belong to the target network. $\Delta = \{((x_1, x_3), r_1), ((x_1, x_4), r_1), ((x_2, x_3), r_1), ((x_2, x_4), r_1), ((x_2, x_5), r_1), ((x_3, x_5), r_1)\}$. Then, PREDICT&ASK assigns to each candidate constraint in Δ a score. We sort the elements of Δ in decreasing order of their score. Suppose that we have the following order $\langle((x_1, x_4), r_1), ((x_2, x_4), r_1), ((x_2, x_5), r_1), ((x_2, x_3), r_1), ((x_3, x_5), r_1), ((x_1, x_3), r_1)\rangle$. Suppose that $\alpha = 1$, which means that we have to exit PREDICT&ASK after one negative answer. We pick the first constraint $((x_1, x_4), r_1)$ in Δ, and we ask the user the recommendation query $AskRec((x_1, x_4), r_1)$, which will be answered yes, as the constraint $((x_1, x_4), r_1)$ belongs to the target network.

The other questions are as follows:

- $AskRec((x_2, x_4), r_1) = yes \quad (\#No = 0)$
- $AskRec((x_2, x_5), r_1) = yes \quad (\#No = 0)$
- $AskRec((x_2, x_3), r_1) = no \quad (\#No = 1 \Rightarrow exit)$

At the end, thanks to PREDICT&ASK three (out of four) constraints are added to the current constraint network (see Figure 1(d)).

5 Prediction Strategies

The way PREDICT&ASK computes the *score* has not been detailed in Section 3. In this section, we present the two techniques that we have used to predict missing constraints. Bessiere et al. (2014) have shown that when a constraint network has some structure, variables of the same given *type* are often involved in constraints with the same relation. Hence, we expect that when variable types are not known in advance, predicting type similarity or type proximity of variables could be done by prediction link techniques.

Link prediction in dynamic graphs is an important research field in data mining. Link prediction can be used for recommendation systems [Li and Chen, 2009], security domain [Krebs, 2002], social networks [Liben-Nowell and Kleinberg, 2003], and many other fields. Several techniques have been proposed in the literature for link prediction. All these techniques compute and assign a score to pairs of nodes $(x,\ y)$, based on the input graph and then produce a ranked list in a decreasing order of scores. They can be viewed as computing a measure of proximity or similarity between nodes x and y, with respect to the network topology. Most of these techniques are based either on node neighborhood or on path ensemble [Lu and Zhou, 2010]. In our experiments we selected one link prediction technique representative of node-neighborhood-based techniques (Adamic/Adar –AA), and one representative of path-ensemble-based techniques (Leicht-Holme-Newman Index –LHN). Both of these techniques have a time complexity in $O(n^3)$. We will see in our experiments that this never takes more than a few milliseconds.

5.1 Adamic-Adar Index (AA)

Adamic and Adar (2003) proposed a measure in the context of deciding when two personal home pages are strongly "related". They compute features of the pages and define the similarity index between two pages to be:

$$\sum_{z:\mathcal{Z}} \frac{1}{log(frequency(z))}$$

where $\mathcal{Z}$ is the set of features shared by x and y. This refines the simple counting of common features by weighting rarer features more heavily. This suggests the measure

$$score(x, y) = \sum_{z \in N(x) \cap N(y)} \frac{1}{log|N(z)|}$$

where $N(x)$ denotes the neighborhood of x, that is, the set of variables with whom it shares a constraint.

5.2 Leicht-Holme-Newman Index (LHN)

Leicht-Holme-Newman (2006) proposed to compute vertex similarity, or proximity, based on the concept that two nodes are similar when their neighbors are similar. This index can be expressed into a matrix form as:

$$S = 2m\lambda_1 D^{-1}(I - \frac{\phi A}{\lambda_1})^{-1} D^{-1}$$

where m is the number of links in the network, λ_1 is the largest Eigenvalue of the adjacency matrix A, D is a diagonal degree matrix, I is the identity matrix, and ϕ $(0 < \phi < 1)$ is a free parameter that assigns higher weights to shorter paths if it is closer to 0 and to longer paths if it is closer to 1 [Lu and Zhou, 2010]. In all our experiments we have set ϕ to 0.5 to assign the same weight to both shorter and longer paths.

6 Experimental Evaluation

We made experiments to evaluate the impact of using PREDICT&ASK in constraint acquisition. We first present the benchmark problems we used for our experiments. Then, we report the results of acquiring these problems with the basic version of QUACQ, with a brute-force algorithm using only recommendation queries (denoted by ONLYREC), and with our P-QUACQ using the Adamic/Adar (AA) and Leicht-Holme-Newman (LNH) indexes to recommend constraints to the user. ONLYREC makes a brute-force use of recommendation queries: it asks recommendation queries on constraints from B and removes redundant constraints from B each time a new constraint is learned, until convergence is reached. Our tests were conducted on an Intel Core i5-3320M CPU @ 2.60GHz $\times$ 4 with 4 Gb of RAM.

6.1 Benchmark Problems

Radio Link Frequency Assignment Problem. The RLFAP is to provide communication channels from limited spectral resources [Cabon *et al.*, 1999]. Here we build a simplified version of RLFAP that consists in distributing all the frequencies available on the base stations of the network. The constraint model has 36 variables with domains of size 36, and 210 binary constraints. We fed QUACQ and P-QUACQ with a basis of 1800 binary constraints taken from a language of 6 arithmetic and distance constraints.

Vessel Loading. Supply vessels transport containers from site to site. The deck area is rectangular. Containers are cuboid, and are laid out in a single layer. All containers are positioned parallel to the sides of the deck. The contents of the containers determine their class. Certain classes of containers are constrained to be separated by minimum distances either along the deck or across the deck. The constraint model has 25 variables with domains of size 25, and 210 binary constraints. We fed QUACQ and P-QUACQ with a basis of 2610 binary constraints taken from a language of 6 arithmetic and distance constraints.

Murder. Someone was murdered last night, and you are summoned to investigate the murder. The objects found on the spot that do not belong to the victim include: a pistol, an umbrella, a cigarette, a diary, and a threatening letter. There are also witnesses who testify that someone had argued with

Table 1: P-QUACQ on RLFAP.

	α	$\#query$	$\#Ask$	$\#AskRec$	$\#no$	$\#yes$
QUACQ	–	1653	1653	–	–	–
ONLYREC	–	1575	–	1575	–	–
P-QUACQ+Random	$+\infty$	964	560	404	322	82
	4	1017	676	341	268	73
	3	1129	817	312	250	62
	2	1281	1013	268	220	48
	1	1553	1370	183	161	22
P-QUACQ+AA	$+\infty$	964	560	404	322	82
	4	970	643	327	250	77
	3	1010	719	291	220	71
	2	1028	784	244	178	66
	1	1229	1055	174	128	46
P-QUACQ+LHN	$+\infty$	964	560	404	322	82
	4	886	564	322	240	82
	3	859	580	279	197	82
	2	**851**	624	227	148	79
	1	1052	878	174	114	60

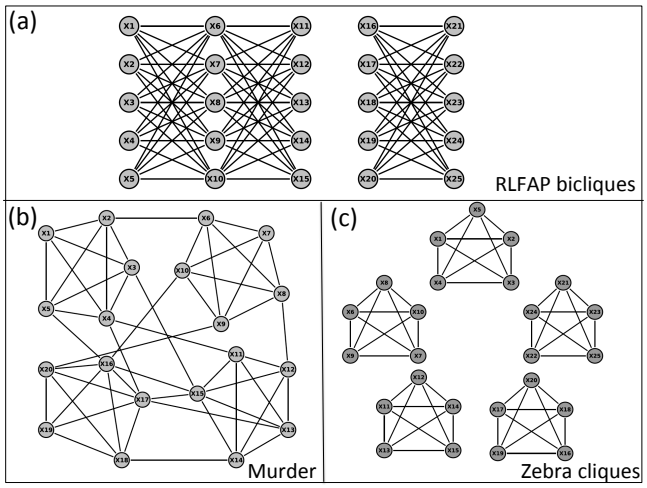

Figure 2: Constraint graphs of our problems.

the victim, someone left the house, someone rang the victim, and some walked past the house several times about the time the murder occurred. The suspects are: Miss Linda Ablaze, Mr. Tom Burner, Ms. Lana Curious, Mrs. Suzie Dulles, and Mr. Jack Evilson. Each suspect has a different motive for the murder, including: being harassed, abandoned, sacked, promotion and hate. Under a set of additional clues given in the description, the problem is who was the Murderer? And what was the motive, the evidence-object, and the activity associated with each suspect. The target network of Murder has 20 variables with domains of size 5, and 53 binary constraints. We fed QUACQ and P-QUACQ with a basis B of 1140 binary constraints based on the language $\Gamma = \{=, \neq, \geq, <, \leq, >\}$.

Zebra problem. The Lewis Carroll's Zebra problem is formulated using 25 variables, with 5 cliques of $\neq$ constraints and 14 additional constraints given in the description of the problem. We fed QUACQ and P-QUACQ with a basis B of 4450 unary and binary constraints taken from a language with 24 basic arithmetic and distance constraints.

6.2 Results

We compare QUACQ, ONLYREC, and P-QUACQ. For P-QUACQ we report results when predicting links with AA or LHN, without cutoff (i.e, $\alpha = +\infty$) and also with four values for the cutoff α (from 1 to 4). We also report results when predicting links with a Random strategy, which serves as baseline selector as it randomly picks a candidate constraint from Δ, and recommends it to the user. For all our experiments we report the number of (standard) queries asked by the basic QUACQ, the number of (recommendation) queries asked by ONLYREC, and the number of queries asked by P-QUACQ. For P-QUACQ we report the number $\#Ask$ of standard queries, the number $\#AskRec$ of recommendation queries, the numbers $\#no$ and $\#yes$ of negative and positive recommendation queries (i.e., $\#AskRec = \#no + \#yes$), and the total number $\#query$ of queries (i.e., $\#query = \#Ask + \#AskRec$). The time overhead of computing scores and generating recommendation queries is not reported because it takes a few milliseconds.

Table 1 reports the results of acquiring the RLFAP problem. We first observe that the number of queries asked by P-QUACQ is always significantly lower than with QUACQ or ONLYREC, whatever the way we predict links in P-QUACQ. We also observe that AA and LHN outperform Random for all values of α, which means that their predictions are correlated to the probability of having a constraint at the selected link. When predicting links with AA, we observe that cutoffs hurt the acquisition: the smaller the cuttoff, the greater the number of queries required for convergence. On the contrary, P-QUACQ+LHN is better with cutoff: it reaches its lower number of queries to learn the RLFAP network when $\alpha = 2$ (851 queries instead of 1653 for basic QUACQ and 1575 for ONLYREC). The good performance of LHN on RLFAP can be explained by the structure of that problem. The RLFAP structure contains bicliques and cliques. The constraints that belong to the same clique can be easily predicted by both the neighborhood-based method, AA, or the path-ensemble-based method, LHN. However, constraints in bicliques cannot be predicted by AA because variables of the same constraint do not share any neighbor (see Figure 2a).

Table 2 reports the results on the Vessel Loading problem. The structure of this problem is quite similar to the structure of RLFAP. Thus, the results follow the same trend as on the RLFAP (P-QUACQ+LHN with $\alpha = 2$ is the best). However, we see that as opposed to the RLFAP, P-QUACQ+AA benefits from the cutoffs.

Table 3 reports the results on the Murder problem. The structure of that problem is essentially composed of cliques, as we can see in Figure 2b. In this case, we observe that P-QUACQ +AA with a cutoff equal to 1 is the best. It requires 367 queries to get the model instead of 585 queries for QUACQ and 1050 for ONLYREC. The good performance of the AA predictor can be explained by the fact that neighborhood-based predictors are effective in detecting cliques.

Table 4 reports the results on the Zebra problem. Again, P-QUACQ with AA or LHN predictors outperforms QUACQ, ONLYREC, and P-QUACQ +Random. When comparing AA to LHN, we observe that, interestingly, P-QUACQ +AA and P-QUACQ +LHN give exactly the same results for all values

Table 2: P-QUACQ on Vessel Loading.

	α	$\#query$	$\#Ask$	$\#AskRec$	$\#no$	$\#yes$
QUACQ	–	2252	2252	–	–	–
ONLYREC	–	2595	–	2595	–	–
P-QUACQ+Random	$+\infty$	1505	864	641	501	140
	4	1655	1159	496	378	118
	3	1686	1212	474	361	113
	2	1758	1368	390	289	101
	1	1934	1658	276	196	80
P-QUACQ+AA	$+\infty$	1505	864	641	501	140
	4	1385	889	496	357	139
	3	1240	852	388	266	122
	2	1195	882	313	194	119
	1	1270	1033	237	128	109
P-QUACQ+LHN	$+\infty$	1505	864	641	501	140
	4	1381	892	489	350	139
	3	1213	831	382	259	123
	2	**1137**	827	310	187	123
	1	1217	988	229	116	113

Table 3: P-QUACQ on Murder.

	α	$\#query$	$\#Ask$	$\#AskRec$	$\#no$	$\#yes$
QUACQ	–	585	585	–	–	–
ONLYREC	–	1050	–	1050	–	–
P-QUACQ+Random	$+\infty$	433	267	166	138	28
	4	453	315	138	115	23
	3	496	380	116	100	16
	2	497	406	91	78	13
	1	523	467	56	49	7
P-QUACQ+AA	$+\infty$	433	267	166	138	28
	4	414	282	132	105	27
	3	404	292	112	86	26
	2	386	301	85	60	25
	1	**367**	313	54	30	24
P-QUACQ+LHN	$+\infty$	433	267	166	138	28
	4	448	309	139	115	24
	3	428	310	118	94	24
	2	439	349	90	70	20
	1	459	401	58	43	15

Table 4: P-QUACQ on Zebra.

	α	$\#query$	$\#Ask$	$\#AskRec$	$\#no$	$\#yes$
QUACQ	–	694	694	–	–	–
ONLYREC	–	4142	–	4142	–	–
P-QUACQ+Random	$+\infty$	675	423	252	221	31
	4	679	496	183	163	20
	3	660	505	155	132	23
	2	711	593	118	106	12
	1	693	625	68	60	8
P-QUACQ+AA	$+\infty$	675	423	252	221	31
	4	602	431	171	141	30
	3	576	434	142	112	30
	2	524	417	107	77	30
	1	**498**	428	70	40	30
P-QUACQ+LHN	$+\infty$	675	423	252	221	31
	4	602	431	171	141	30
	3	576	434	142	112	30
	2	524	417	107	77	30
	1	**498**	428	70	40	30

of the cutoff α. This can be explained by the fact that only the relation $\neq$ shows a structure in the network, and that structure is such that all cliques are isolated, as illustrated in Figure 2c. Such a structure is perfectly well detected both by AA and LHN. This explains that they behave the same and that the shorter the cutoff, the better.

This experimental analysis clearly shows that the use of prediction strategies with recommendation queries can significantly reduce the number of queries asked to the user. The brute-force use of recommendation queries (ONLYREC) is always close to the worst case (i.e., close to $|B|$ queries). The AA prediction strategy seems to be particularly well-suited to problem containing cliques of constraints, whereas the LHN can be highly efficient to predict biclique structures.

7 Related Work

Several papers have already proposed to use the structure of the constraint graph to decrease the number of examples needed to learn the target constraint network. Beldiceanu and Simonis (2012) have proposed MODELSEEKER, a passive constraint acquisition system devoted to problems having a regular structure. MODELSEEKER learns global constraints from the global constraints catalog ([Beldiceanu *et al.*, 2007]) whose scopes are the rows, the columns, or any other structural property MODELSEEKER can capture. The counterpart is that it misses any constraint that does not belong to one of the structural patterns it is able to capture. Bessiere et al. (2014) introduced a new concept of query, called generalization query. By using some background knowledge, namely types of variables, a generalization query asks the user whether or not a learned constraint can be generalized to other scopes of variables of the same types as those of the learned constraint. The drawback of such queries is that they require types of variables to be provided by the user. To overcome this weakness, Daoudi et al. (2015) have proposed to learn types of variables during the constraint acquisition process, and then to use the learned types to generate generalization queries. The advantage of such an approach is that there is no need for the user to provide the types. Of course learning the types requires extra queries that were not needed when types are given for free at the beginning of the learning process. In addition, generalization queries do not work on all problems. They work only for the problems for which variables can be grouped into types.

By contrast, in our work, recommendation queries are generic and do not require any background knowledge to be generated. By using techniques borrowed from link prediction in dynamic graphs, we infer constraints that are more likely to belong to the target constraint network, and that are validated by asking recommendation queries to the user.

8 Conclusion

We have proposed a new kind of queries, called recommendation queries. To deal with these queries, we have proposed a generic constraint recommender algorithm, PREDICT&ASK, which uses techniques borrowed from link prediction to predict constraints that are likely to belong to the target network. Finally, we have plugged PREDICT&ASK into QUACQ to have a boosted version called P-QUACQ. Our experiments on several benchmark problems show that our new technique outperforms the basic QUACQ. An interesting direction would be to use a reinforcement learning to decide on the use of neighborhood-based predictions or path-ensemble-

based predictions.

References

[Adamic and Adar, 2003] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[Beldiceanu and Simonis, 2012] Nicolas Beldiceanu and Helmut Simonis. A model seeker: Extracting global constraint models from positive examples. In *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 141–157, 2012.

[Beldiceanu *et al.*, 2007] Nicolas Beldiceanu, Mats Carlsson, Sophie Demassey, and Thierry Petit. Global constraint catalogue: Past, present and future. *Constraints*, 12(1):21–62, 2007.

[Bessiere *et al.*, 2005] Christian Bessiere, Remi Coletta, Frédéric Koriche, and Barry O'Sullivan. A sat-based version space algorithm for acquiring constraint satisfaction problems. In *Machine Learning: ECML 2005, 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005, Proceedings*, pages 23–34, 2005.

[Bessiere *et al.*, 2007] Christian Bessiere, Remi Coletta, Barry O'Sullivan, and Mathias Paulin. Query-driven constraint acquisition. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 50–55, 2007.

[Bessiere *et al.*, 2013] Christian Bessiere, Remi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Constraint acquisition via partial queries. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013.

[Bessiere *et al.*, 2014] Christian Bessiere, Remi Coletta, Abderrazak Daoudi, Nadjib Lazaar, Younes Mechqrane, and El-Houssine Bouyakhf. Boosting constraint acquisition via generalization queries. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pages 99–104, 2014.

[Bessiere *et al.*, 2015] Christian Bessiere, Frédéric Koriche, Nadjib Lazaar, and Barry O'Sullivan. Constraint acquisition. *Artificial Intelligence, In Press*, 2015.

[Cabon *et al.*, 1999] Bertrand Cabon, Simon de Givry, Lionel Lobjois, Thomas Schiex, and Joost P. Warners. Radio link frequency assignment. *Constraints*, 4(1):79–89, 1999.

[Daoudi *et al.*, 2015] Abderrazak Daoudi, Nadjib Lazaar, Younes Mechqrane, Christian Bessiere, and El-Houssine Bouyakhf. Detecting types of variables for generalization in constraint acquisition. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, pages 413–420, 2015.

[Freuder, 1999] Eugene C. Freuder. Modeling: The final frontier. In *1st International Conference on the Practical Applications of Constraint Technologies and Logic Programming*, pages 15–21, London, UK, 1999. Invited Talk.

[Krebs, 2002] Valdis E. Krebs. Uncloaking terrorist networks. *First Monday*, 7(4), 2002.

[Lallouet *et al.*, 2010] Arnaud Lallouet, Matthieu Lopez, Lionel Martin, and Christel Vrain. On learning constraint problems. In *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, 27-29 October 2010 - Volume 1*, pages 45–52, 2010.

[Leicht *et al.*, 2006] Elizabeth Leicht, Petter Holme, and Mark Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[Li and Chen, 2009] Xin Li and Hsinchun Chen. Recommendation as link prediction: a graph kernel-based machine learning approach. In *Proceedings of the 2009 Joint International Conference on Digital Libraries, JCDL 2009, Austin, TX, USA, June 15-19, 2009*, pages 213–216, 2009.

[Liben-Nowell and Kleinberg, 2003] David Liben-Nowell and Jon M. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pages 556–559, 2003.

[Lu and Zhou, 2010] Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *CoRR*, abs/1010.0725, 2010.

[Shchekotykhin and Friedrich, 2009] K.M. Shchekotykhin and G. Friedrich. Argumentation based constraint acquisition. In *Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM'09)*, pages 476–482, Miami, Florida, 2009.